AFIT/GCE/ENG/92D-09

A RADIOSITY APPROACH TO REALISTIC IMAGE SYNTHESIS

THESIS

Richard L. Remington
Captain, USAF

AFIT/GCE/ENG/92D-09

AFIT/GCE/ENG/92D-09

A RADIOSITY APPROACH TO REALISTIC IMAGE SYNTHESIS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Computer Engineering

Richard L. Remington, B.S.C.E.

Captain, USAF

December, 1992

Approved for public release; distribution unlimited

Preface

The purpose of this study was to design and implement a state-of-the-art realistic image synthesis system. A primary goal was keeping the final image as view independent as possible and to bring the Air Force Institute of Technology (AFIT) up to date in this area.

I am indebted to several people for their assistance with this project. I would like to thank 2nd Lt. Rob Pittman for his assistance in creating the input geometry file for the AWACS aircraft interior. Without his assistance, a good model for the diffuse radiosity implementation would not have been possible. I would also like to thank Dr. Kirk Mathews for keeping me on track with the actual physics of light transport, and providing the equations for the specular radiosity calculations. I also want to thank my thesis advisor, LtCol Phil Amburn, for his guidance and support. He was always available to work through my problems and his questions and comments made this research possible.

I especially wish to thank my wife, Jeanette, for her patience, understanding, and unending support.

Richard L. Remington

Table of Contents

		Page
Preface		ii
Table of Contents		iii
List of Figures		vii
Abstract		ix
I.	Problem Statement	1-1
	1.1 Background	1-1
	1.2 Summary of Current Knowledge	1-1
	1.3 Problem	1-2
	1.4 Scope	1-3
	1.5 Approach/Methodology	1-3
	1.6 Overview of Thesis	1-4
II.	Literature Review	2-1
	2.1 Introduction	2-1
	2.2 Ray Tracing	2-1
	2.3 Distributed Ray Tracing	2-2
	2.4 Radiosity	2-3
	2.4.1 Form Factor Calculation	2-5
	2.4.2 Adaptive Subdivision	2-6
	2.4.3 Rapid Hierarchical Radiosity	2-7
	2.4.4 Progressive Refinement	2-8
	2.4.5 Progressive Refinement with Ray Tracing for Form Factors	2-8

Page

2.4.6 Meshing Algorithms 2-9

2.4.7 Radiosity Conclusions 2-13

2.5 Extensions 2-13

2.5.1 The Rendering Equation 2-13

2.5.2 Particle Transport 2-14

2.5.3 Extended Radiosity 2-14

2.6 Multi Pass Methods 2-15

2.6.1 Wallace, Cohen and Greenberg 2-15

2.6.2 Sillion and Puech 2-16

2.6.3 Chen, Rushmeier, Miller and Turner 2-16

2.6.4 Sillion, Arvo, Westin, and Greenberg 2-17

2.7 Conclusion 2-18

III. Diffuse Radiosity Design 3-1

3.1 Introduction 3-1

3.2 Starting Point: Basic Progressive Refinement Radiosity . . . 3-1

3.2.1 Capabilities 3-2

3.2.2 Comments 3-2

3.3 Automated Meshing 3-3

3.4 Data Structures 3-4

3.5 Input Files 3-7

3.6 Ray Traced Form Factors 3-8

3.7 Explanation of Algorithm 3-9

3.8 Viewing the Image 3-12

3.9 Conclusion 3-13

Page

IV. Directional Diffuse Radiosity Design 4-1

4.1 Introduction 4-1

4.2 Background and Overview 4-1

4.3 Representing BRDFs 4-2

4.3.1 Data Structures for BRDFs 4-4

4.4 Representing Intensities 4-4

4.4.1 Data Structures for Intensities 4-6

4.5 Calculating Radiosity Distributions 4-7

4.5.1 Explanation of the Algorithm 4-7

4.6 Viewing the Image 4-11

4.7 Conclusion 4-11

V. Results 5-1

5.1 Progressive Radiosity 5-1

5.1.1 Automatic Meshing 5-1

5.1.2 Progressive Refinement 5-1

5.1.3 Viewing the Image 5-7

5.2 Directional Diffuse Radiosity 5-7

5.3 Conclusion 5-12

VI. Recommendations for Future Research 6-1

6.1 Meshing Algorithms 6-1

6.2 Source Sampling 6-2

6.3 Bi-directional Reflectance Distribution Functions 6-2

6.4 Ray Tracing 6-2

6.5 Other Enhancements 6-3

6.6 Conclusions 6-3

Appendix A. Format for Display Parameter File A-1

Page

Appendix B. Format for Geometry File B-1

Appendix C. Using radview C-1

C.1 Mouse and Keyboard Inputs C-1

Appendix D. Using *prorad* D-1

Appendix E. Mathematica Program Listing E-1

Appendix F. Reflectance File Format F-1

Appendix G. Expanded BRDF Derivation G-1

Appendix H. Using *extrad* H-1

Bibliography BIB-1

Vita VITA-1

List of Figures

Figure Page

2.1. A ray-traced picture 2-2
2.2. Projecting patches into hemi-cube. 2-6
2.3. Casting rays from point to determine form factors. 2-7
2.4. Shortcomings of hemi-cube approach. 2-9
2.5. Tracing rays from vertices. 2-10
2.6. Shadows do not affect meshing with uniform mesh. 2-11
2.7. Shadows affect meshing with quad-tree mesh. 2-12
2.8. Scene efficiently meshed with bsp-tree mesh. 2-12
2.9. Rendering Equation. 2-14

3.1. Original Cornell Room 3-3
3.2. Data structure used for patches, elements, and vertices 3-5
3.3. T-Vertices 3-6
3.4. Image showing black dots where pixel drop occurred 3-7
3.5. Arrangement of θ_1, θ_2, and r 3-9
3.6. Psuedocode for radiosity program 3-10

4.1. Angles and vectors used in reflectance equation. 4-2
4.2. Plot of $|ReY_{0,0}|$. 4-4
4.3. Plot of $|ReY_{3,0}|$. 4-5
4.4. Plot of $|ReY_{4,1}|$. 4-5
4.5. Plot of $|ReY_{6,3}|$. 4-6
4.6. Data structure used for representing intensities. 4-7
4.7. Psuedocode for extended radiosity implementation. 4-8
4.8. Angles used for intensity calculations. 4-9

Figure Page

5.1. Cornell room showing initial patches. 5-2
5.2. Cornell room showing subdivision after 1 level. 5-2
5.3. Cornell room showing subdivision after 2 levels. 5-3
5.4. Cornell room showing subdivision after 3 levels. 5-3
5.5. Cornell room showing subdivision after 4 levels. 5-4
5.6. Cornell room showing subdivision after 5 levels. 5-4
5.7. Cornell room showing subdivision after 6 levels. 5-5
5.8. Cornell room showing subdivision after 7 levels. 5-5
5.9. Cornell room showing subdivision after 8 levels. 5-6
5.10. Cornell room showing final subdivision of elements. 5-6
5.11. Cornell Room after one shooting patch. 5-7
5.12. Cornell Room after two shooting patches. 5-8
5.13. Cornell Room after three shooting patches. 5-8
5.14. Cornell Room after ten shooting patches. 5-9
5.15. Cornell Room after convergence (63 patches shot). 5-9
5.16. Interior of AWACs aircraft as rendered by *prorad.* 5-10
5.17. Closer look at two operator consoles 5-10
5.18. Details of AWACs chairs 5-11
5.19. Commanders console 5-11
5.20. Image showing slight specular highlights($\alpha = .15$) from front (viewer is in direction of reflection). 5-12
5.21. Image showing slight specular highlights($\alpha = .15$) from back (viewer is at light source). 5-13
5.22. Image showing slight specular highlights($\alpha = .15$) from side. 5-13
5.23. Image showing tighter specular highlights ($\alpha = .10$) from front. 5-14
5.24. Image showing tighter specular highlights ($\alpha = .10$) from back. 5-14
5.25. Image showing tighter specular highlights ($\alpha = .10$) from side. 5-15

G.1. Angles and vectors used in reflectance equation. G-2

Abstract

The Air Force Institute of Technology (AFIT) has not had the capability to generate photorealistic images. This thesis is a research effort to implement such a system at AFIT, and to apply knowledge from the particle transport community to this problem.

Investigation into various techniques for generating photorealistic images led to the conclusion that the radiosity method would serve as the foundation for development. This is because of the increased accuracy of the diffuse reflections with the radiosity method and the fact that the images produced with radiosity are view-independent.

Two distinct tools to generate photorealistic images are described. The first of these follows the progressive radiosity methodology using ray tracing for form factor calculation. This tool models only diffuse reflections. The second tool is also based on the radiosity method, but does away with the restriction that all surfaces be lambertian diffuse and models specular reflections as well.

The design and implementation of the specular radiosity tool uses continuous functions to represent the ***bidirectional reflectance distribution functions***, (BRDFs) and intensities. A new method of representing the BRDFs using a truncated Legendre polynomial expansion is presented.

A RADIOSITY APPROACH TO REALISTIC IMAGE SYNTHESIS

I. Problem Statement

1.1 Background

For years, researchers have been trying to create realistic looking pictures with computer graphics programs. In the early years, all of the work was done with direct reflection models. With these techniques, the only thing taken into consideration was the interaction of the light sources with the surface in question. As these approaches matured, techniques to add shadows were incorporated. This made the images produced more realistic, but it was still obvious that they were computer generated. We have long known that direct reflection models will not do an adequate job of solving the global illumination problem (26, 12). The intensity of light at any point in a scene depends on the intensities of light at all other points in the scene. Early work on this problem made strong assumptions about the nature of reflected light, that it was either purely specular or purely diffuse. Obviously, this does not coincide with reality (28). Recently, techniques have been introduced to relax these assumptions (28, 27, 6). These techniques were investigated in this research.

1.2 Summary of Current Knowledge

Work in the area of realistic image synthesis began with work on empirical shading models at the University of Utah in the early 70's. In the early 80's, realism greatly increased with the introduction of ray-tracing techniques. These modelled specular reflections and refraction far better than previous techniques. Shortly after that, researchers at Cornell University introduced radiosity to the computer graphics community. The radiosity technique complemented ray-tracing in that it ignored the specular terms but modelled

the diffuse reflections accurately. Both of these techniques were important, but neither modelled all types of reflected light. Ray tracing methods can be modified to more accurately model the light interactions, but these techniques "require huge numbers of rays to be cast per pixel to avoid perceptible noise in the image" (6:165). Radiosity has been extended to model specular reflections (21), but such techniques require immense storage and computation time, rendering it impractical (28).

In 1987 came the next logical approach, use both radiosity and ray-tracing to take advantage of the inherent strengths of both (31). This technique proved to work quite well, except that by calculating diffuse terms with radiosity and specular terms with ray-tracing, the two types of reflection are considered independent, which is not the case. To overcome this, researchers have developed techniques to consider specular and diffuse reflections in the radiosity step to get accurate diffuse terms that take specular reflections into account, and then do a ray-tracing step to refine and get more accurate specular terms based on all of the knowledge gained in the extended radiosity step. (28, 27, 6)

1.3 Problem

The purpose of this research was to design and implement a state-of-the-art realistic image generation system utilizing the latest advances in computer graphics research. In doing this, all mechanisms of light transport were considered. These are: diffuse to diffuse, specular to diffuse, diffuse to specular, and specular to specular (31). Since there are more than two surfaces in a normal scene, complex combinations of these mechanisms need to be considered as well.

The primary focus was to create as realistic an image as possible in a view independent manner. This gives the added benefit of being able to move about in the scene without having to recompute the lighting.

The secondary focus was to make the view independent step faster. Traditionally,

when high quality images are desired, long computation times were expected. As new techniques are developed, these images can be produced in shorter and shorter times. This research combines some of the most promising speed up techniques introduced to date.

1.4 Scope

This research focuses primarily on developing a progressive radiosity renderer with extended form factors. This implementation allows for very accurate diffuse light representation as well as specular reflections in a view independent manner. A method of viewing that does not require ray tracing was also investigated.

1.5 Approach/Methodology

This research project was developed under the evolutionary life-cycle development methodology (37). With this method, a basic capability was developed right away and enhancements came in phases, each one furthering the capability and still providing a working version of the software. The phases in this development were as follows:

1. Develop a basic progressive refinement radiosity program. This software is a modified version of the program presented in *Graphics Gems II* by Eric Chen (2). The modifications include allowing the user to specify the input geometry files, and adding a procedure to save the output image to a file so it could be viewed at a later time. The Graphics Gems code did not have any drawing routines, so these had to be added as well. This implementation used uniform meshing and a hemi-cube to calculate form factors. All polygons were flat shaded.
2. The next phase involved converting the hemi-cube form factor approach to using ray-traced form factors. This method calculates the radiosity at the vertices instead of the patch centers. This makes it very easy to display a continuously shaded image, as Gouraud shading does this in hardware. Data structures were modified

to take advantage of the new algorithms to achieve an impressive speed up. The breaking of patches into elements had to be done manually in the *Graphics Gems* code. Procedures were added to automate this. This produced a full-featured, diffuse radiosity implementation which was kept separate as a complete product.

3. The final step involved expanding the diffuse progressive refinement radiosity program to include specular reflection contributions. This was done following a technique using spherical harmonic expansion introduced by Sillion et al. at SIGGRAPH '91 (27). At this point the image that is generated is view independent. This is of significant advantage in that it can be used with an interactive viewer to view the scene from all directions.

1.6 Overview of Thesis

The following chapter describes some of the most important work to date in the area of realistic image synthesis. Chapter 3 describes the design of the diffuse radiosity based realistic image synthesis system. Chapter 4 describes the changes and extensions to the design to implement the extended radiosity which takes all but the perfectly specular effects into account. Results of this research are discussed in Chapter 5, and future research areas and conclusions are discussed in Chapter 6.

II. Literature Review

2.1 Introduction

This chapter is a survey of some of the more important work in the area of global illumination, with primary focus on the radiosity method. The first question that needs to be addressed is what is meant by a realistic image? Hall and Greenberg write: "Our goal in realistic image synthesis is to generate an image which evokes from the visual system a response indistinguishable from the actual environment" (17:10). Obviously, this is very difficult to achieve. This chapter will attempt to introduce the reader to the major accomplishments in this area to date. Since this is such a broad area, the focus will be non-interactive applications, where fast image generation is not as important as achieving a high degree of realism.

2.2 Ray Tracing

Some of the most realistic images to date have been produced by a technique known as ray tracing (35). Instead of trying to find the illumination for each surface as in previous methods, ray tracing calculates the illumination for each pixel. This is done by tracing a ray from the eye, through a point on the image plane until it comes in contact with a surface. The direct illumination of the surface at this point is then calculated. If the surface is reflective or transparent, additional rays must be traced from the intersection point in the direction of reflection or refraction to determine their contribution to the illumination at this point. Notice that this automatically takes care of the hidden surface problem because the value is only calculated for the first surface hit. (20:661-669)

The ray tracing method was the first to attempt to model the global illumination of a scene. The primary strength of this method is its ability to handle shadows, specular reflections, and transparency, (Figure 2.1). Unfortunately, ambient and diffuse light are not handled accurately. Basic ray tracing allows for no color bleeding between surfaces due

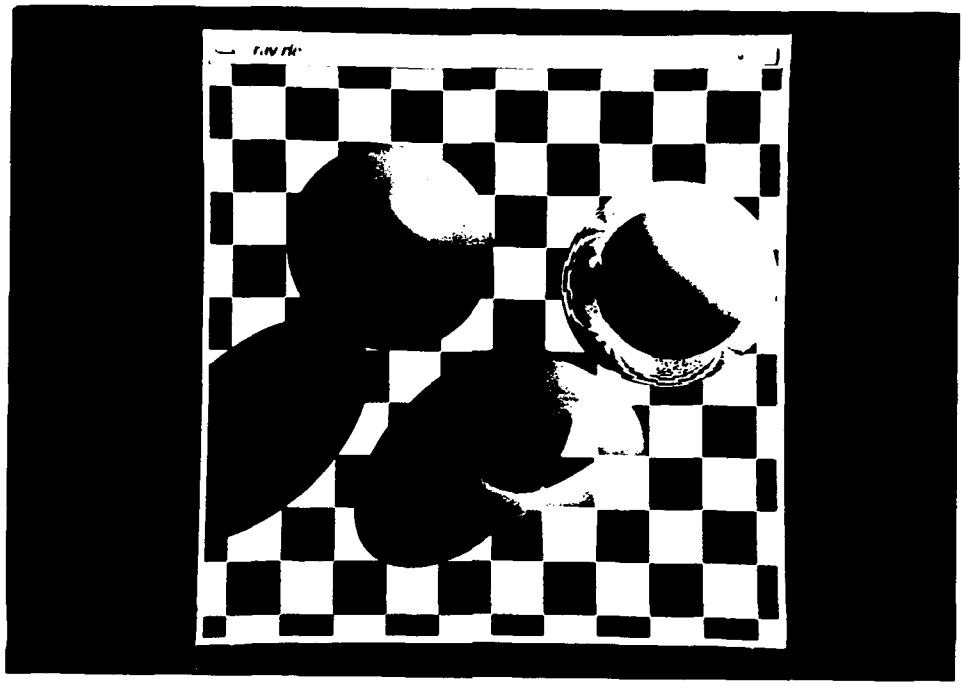

Figure 2.1. A ray-traced picture: note the sharp shadows, reflections and transparency.

to diffuse reflection. Ray tracing is also computationally expensive, often taking several hours to compute a single image. (14:776-784)

2.3 Distributed Ray Tracing

Because standard ray tracing uses infinitesimal rays to do its sampling across a regular grid, there are problems with aliasing. Distributed ray tracing attempts to reduce this by taking a stochastic approach to supersampling that replaces aliasing with random noise, which is much less noticeable to humans (11). Instead of sampling a regular grid, a statistical distribution is used for the rays. This same approach can be used to model various special effects by applying the same basic technique of stochastic sampling to distribute the rays over time, space, etc. With these techniques one can achieve motion blur, depth of field, soft shadows, etc. (14:788-792)

While the distributed ray tracing method does take care of some of the problems associated with standard ray tracing, it still does not properly handle ambient lighting or

diffuse reflections (22), and the fact that it uses supersampling greatly increases the time required to generate an image (14:790).

2.4 Radiosity

Radiosity was introduced to the computer graphics community in 1984 (16), and has as its foundation the field of radiative heat transfer. In this method, the basic principles of conservation of energy are applied to a closed environment. All energy reflected from or emitted by every surface is accounted for by its reflection from or absorption by other surfaces.

The goal of the radiosity method is to calculate the light leaving a surface (its radiosity). Radiosity is the total rate of energy leaving a surface. (energy/unit time/unit area)

To use the radiosity method, a scene is broken up into small discrete planar areas called patches. The radiosity of any patch is assumed constant, so that must be considered when deciding when to break polygons into patches. Once this is done, the radiosity of each patch is calculated by determining the illumination of the patch by the patch in question. This radiosity depends on any light reflected plus light emitted. Thus, the basic radiosity equation is:

$$B_i = E_i + \rho_i \sum_j B_j F_{i,j} \tag{2.1}$$

Where:

B_i — Radiosity of patch i

E_i — Emission from patch i (0 for non light sources)

ρ_i — Reflectivity of patch i

$F_{i,j}$ — Form Factor from patch i to patch j

which can be rewritten in matrix form:

$$\begin{pmatrix} 1 & -\rho_1 F_{1,2} & \cdots & -\rho_1 F_{1,N-1} & -\rho_1 F_{1,N} \\ -\rho_2 F_{2,1} & 1 & \cdots & -\rho_2 F_{2,N-1} & -\rho_2 F_{2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\rho_{N-1} F_{N-1,1} & -\rho_{N-1} F_{N-1,2} & \cdots & 1 & -\rho_{N-1} F_{N-1,N} \\ -\rho_N F_{N,1} & -\rho_N F_{N,2} & \cdots & -\rho_N F_{N,N-1} & 1 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_{N-1} \\ B_N \end{pmatrix} = \begin{pmatrix} E_1 \\ E_2 \\ \vdots \\ E_{N-1} \\ E_N \end{pmatrix} \quad (2.2)$$

Once the patches are defined, the form factors must be calculated. There are two form factors for every pair of patches. These form factors define the fraction of energy leaving one patch and arriving at the other. They are purely a function of geometry between the patches, and are thus view independent. For even relatively simple environments, these form factors become the most computationally expensive part of the radiosity computation.

The equation for the form factors is as follows:

$$F_{i,j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j dA_i \quad (2.3)$$

This equation for the form factors is a complicated double integral over the two patches involved. There is no analytical solution to this integral in general, so some approximation method must be used. These are discussed in the next section.

Once the form factors are calculated, there is a series of n simultaneous equations with n unknowns (n being the total number of patches). There are many known techniques for solving these equations, but it appears that the Guass-Siedel iterative approach is especially well suited for this case, given that the matrix is diagonally dominant (9). The solution is usually found within five or six iterations.

With the matrices solved, the radiosities for each patch are known. These values must be used to calculate the radiosities at each vertex for rendering. Bilinear interpolation within each patch is one way to accomplish this. (7:1-7)

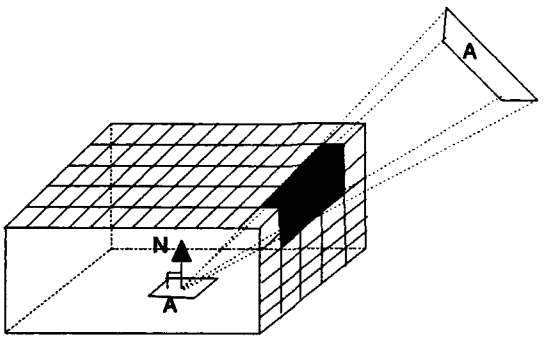

Figure 2.2. Projecting patches into hemi-cube.

2.4.1 Form Factor Calculation

2.4.1.1 Stokes Theorem The initial solution was to take a numerical approach by converting the double area integral into a double contour integral using Stoke's Theorem (15:795-797). This method only works for non-occluded environments and is very computationally expensive (9).

2.4.1.2 Hemi-Cube In 1985, another technique for calculating form factors was developed which is computationally much simpler and can handle complex environments (9). It is based on the Nusselt Analog and involves placing half a cube (a hemi-cube) around the patch in question. Each side of this hemi-cube is subdivided into small grid cells (50-100 per side). Before any calculations can be done using the hemi-cube, the contribution of each grid cell to the overall form factor is calculated and stored with that cell. These are called *delta form factors* and are used to calculate the form-factors. Now each other patch in the environment is projected through this hemi-cube to the patch in question (Figure 2.2). Hidden surfaces are determined by standard Z buffer techniques. Now, the ID of the projected patch is stored in the grid cell of the hemi-cube that is intersected.

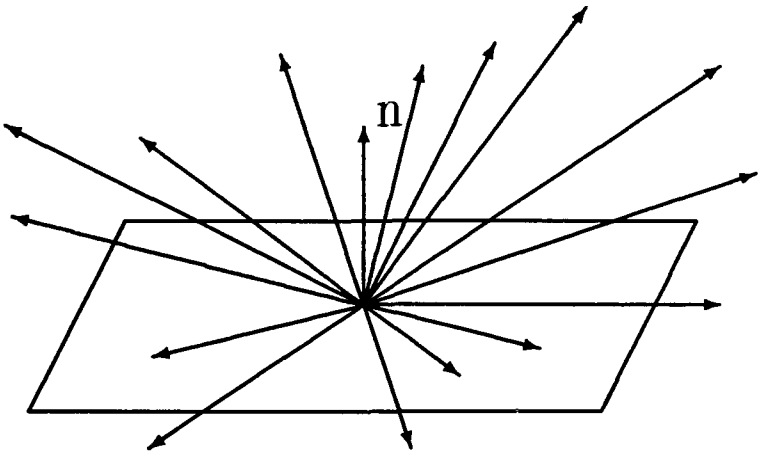

Figure 2.3. Casting rays from point to determine form factors.

Each patch's form factor is determined by summing the pre-computed delta form factors for each grid cell containing the patch's ID. The hemi-cube is then set around the next patch. A row of form factors is created for each hemi-cube location.

There are a number of problems with the hemi-cube method. While it is true that the hemi-cube made radiosity a tractable method, it does this by discretizing the angles under consideration. This can create aliasing problems, leading to inaccuracies in form factor calculations. Another problem is the fact that some surfaces may fall between the hemi-cube pixels, thus not being taken into consideration when calculating the proper intensities. These problems can be lessened by increasing the hemi-cube resolution, but this increases the complexity greatly and cannot be considered with complex environments. (32:317)

2.4.1.3 Ray Casting to Calculate Form Factors In 1986, an alternative method for calculating the form factors was introduced (25). Instead of using a hemi-cube as described above, this method uses ray casting to calculate the form factors. This is done by tracing rays outward from the surface in directions distributed over the entire hemisphere above the surface (Figure 2.3). This gives the calculation the flexibility of non-uniform sampling. This approach tends to be more accurate than the hemi-cube method, but since

most of the hemi-cube calculations can be done with z-buffer hardware, the hemi-cube method is more widely used (7:5).

2.4.2 Adaptive Subdivision Up until now, the assumption has been made that the radiosity is continuous across an entire patch. This works when radiosities vary little or if patches are very small. As stated earlier, making smaller patches is not always an acceptable solution (32).

To overcome these problems, a two level hierarchy of subdividing an environment was adopted (10). In this method, patches with a high radiosity gradient are subdivided into smaller elements which can again be subdivided until an acceptable gradient is achieved. A form factor is now required from each element to each patch, which does increase the number of form factors that needs to be stored. This is better than increasing the number of patches though, because only patches are used as sources of radiosity. The elements are merely receivers of the radiosity. This increases the size of the matrix to m by n, where m is the number of elements, instead of m by m as would be the result if the number of patches was increased to m.

2.4.3 Rapid Hierarchical Radiosity In 1991, another method of breaking up large patches was introduced (18). The algorithm constructs a hierarchical representation of the form factor matrix by adaptively subdividing patches into sub-patches according to a user supplied error bound. The primary result of this hierarchical subdivision is that the decomposed form factor matrix has at most $O(n)$ blocks. This subdivision also insures that all form factors are generated to the same precision, thus removing many common image artifacts due to inaccurate form factors.

The algorithm is based on methods developed for solving the N-body problem. It uses a recursive refinement approach to decompose a polygon into patches and elements and to build a hierarchical representation of the form factor matrix. Unfortunately, the

methods presented are not universal, they work best for relatively large polygons with high brightness gradients. Given these constraints, the results of this method are impressive, rendering fairly complex scenes in a matter of seconds.

2.4.4 Progressive Refinement The major problems with the radiosity methods are the time and memory required to produce images. To answer this problem the radiosity process was reversed (8). Instead of the traditional way of determining the contribution of all other patches to the patch in question, the process is reversed and the contribution of the current patch to all of the other patches is now calculated. In this method, the columns of the form factor matrix are filled in instead of rows. This allows the image to be displayed during the calculation process, and the entire image grows in intensity as the effects of more and more patches are shown. Theoretically, this method is no faster at generating an image than the traditional method, but if the patches are ordered (using those patches which radiate the most energy first), this method converges on an acceptable solution quickly. This is because a small number of light sources and bright surfaces determine most of the global illumination in an environment. This method also requires much less memory usage. Instead of having to store the large n by n array of form factors, the form factors are calculated for each iteration. This reduces storage costs from $O(n^2)$ to $O(n)$.

2.4.5 Progressive Refinement with Ray Tracing for Form Factors In 1989, an extension to the basic progressive refinement radiosity was introduced which did away with the sampling problems inherent in the hemi-cube algorithm for calculating form factors (32). This approach uses ray tracing to perform the numerical integration of the form factor equation. Instead of shooting light out from the source in the uniform directions determined by the grid on the hemi-cube, the illumination source is sampled from the point of view of each of the other surfaces in the environment. Illumination is determined at exactly those points for which shading is desired (the vertices), thus reducing problems with inadequate sampling (Figures 2.4 and 2.5).

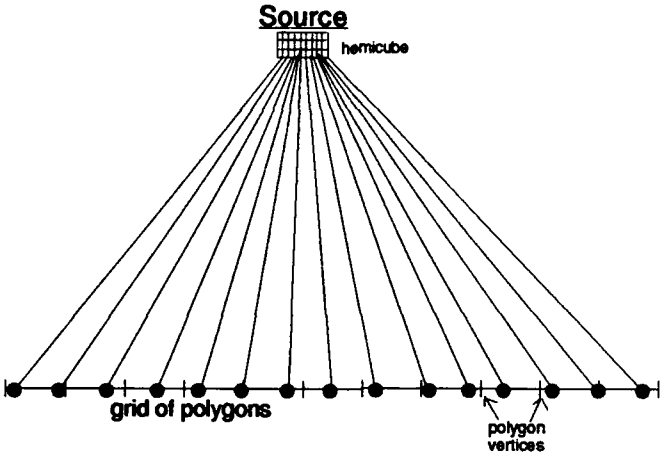

Figure 2.4. Shortcomings of hemi-cube approach, note how some patches get hit once some twice and some may get none at all.

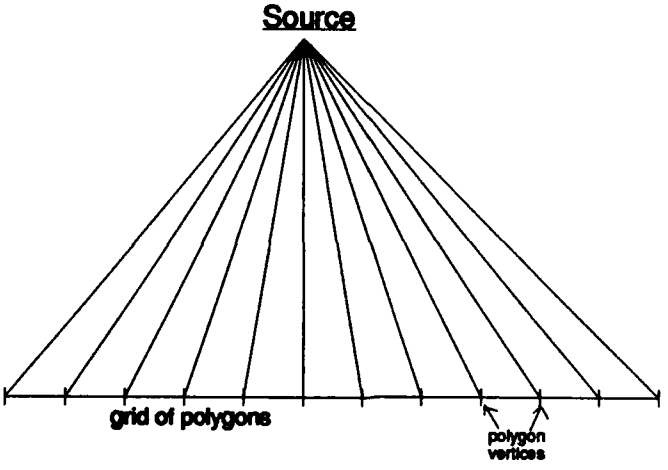

Figure 2.5. Tracing rays from vertices assures each vertex hit.

The basic algorithm is as follows: Find the brightest patch (source). "Instead of performing a hemi-cube at this source, each element vertex in the scene is then visited and a form-factor is computed from the source surface to the vertex by shooting rays from the vertex to sample points on the source. ... The development of this technique was driven by the constraint that it provide good, approximate results for a low number of rays, while allowing increasingly accurate results for higher numbers of rays." (32:318)

2.4.6 Meshing Algorithms One of the most difficult problems addressed with radiosity is meshing the environment. The basic problem here is that to pick up on some of the subtle lighting effects, the mesh needs to be very fine in some areas, but the finer the mesh the more elements there are and the more work needs to be done to generate an image.

Several techniques have been used to do the dividing of polygons into patches and elements. The most straightforward method that does have some merit is to have the user specify for each polygon how it should be broken into patches and elements. Then the meshing is uniform within the polygon. This has the advantage of being simple (Figure 2.6), but has a number of problems. These stem from the fact that the user usually does not know how far a given polygon should be divided. This results in not having a fine enough mesh to properly determine shadow boundaries or having an unnecessarily fine mesh where very little is happening. (30)

To overcome some of the problems with uniform meshing, one would like to have small elements where the radiosity changes rapidly, and larger elements everywhere else. Ideally, the faster the radiosity changes, the smaller the elements. Adaptive subdivision is one way to accomplish this. It works by starting with an initial mesh and calculating the radiosity at each vertex. Any edges where the radiosity varies by too much are subdivided and the process is repeated. One common way of implementing this is to use a quad-tree data structure to represent the mesh, dividing each element into four sub-elements when

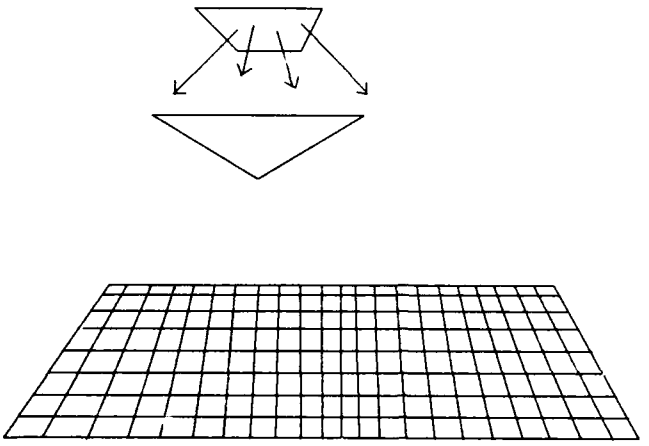

Figure 2.6. Shadows do not affect meshing with uniform mesh.

the radiosity gradient is too high (Figure 2.7). While the quad-tree structure makes for a relatively straightforward implementation, it can result in some visible artifacts. Most of these are caused from the fact that when a quad-tree is divided non-uniformly, T-vertices are a result. T-vertices cause problems with the interpolations for Gouraud shading and also can also result in pixel drop. Pixel drop occurs at t-vertices because the new vertex may not exactly match the edge of the adjoining polygon. The result of this is some pixels do not get colored. The uniformity of the quad-tree implementation can also result in a large number of elements, although certainly lower than with uniform meshing. (30)

Ideally, instead of using perfect quadrilaterals to represent the scene and making them smaller and smaller along areas of high radiosity gradient, the elements should be divided along lines of similar visibility. A method that does this is known as *discontinuity meshing.* Since the lines of similar visibility rarely lie along the lines of a uniform mesh, nonuniform meshes work best with discontinuity meshing. BSP (Binary Space Partitioning) trees are a good choice of data structure for this method (5, 23). The main advantage is the fact that the splitting planes are placed on lines of discontinuity. Instead of having to represent a diagonal sharp shadow with hundreds of quadrilaterals as with the quad-tree method, a

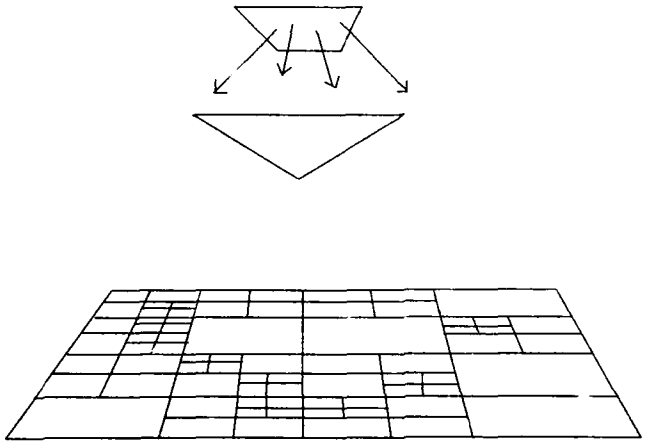

Figure 2.7. Shadows affect meshing with quad-tree mesh.

BSP tree can do it with one (Figure 2.8). Unfortunately, this method still suffers from the generation of T-vertices and finding the lines of discontinuity can be very difficult.

2.4.7 Radiosity Conclusions The radiosity method calculates the lighting of a scene in a completely view-independent way. This allows for very accurate modeling of diffuse reflection, including the phenomena of "color bleeding", where the color from one surface is diffusely reflected onto another. It also does away with the directionless ambient-lighting term used in ray tracing. Ambient light is now accurately modelled. (9:31)

Unfortunately, this method does not account for specular reflection or refraction at all. Due to the directional nature of these properties, they can not be accurately modelled with this approach (21). Adequate meshing of the environment is also a big problem with radiosity. Simple meshing algorithms either do not accurately represent the lighting or use far too many polygons for efficient calculations (4), and more complex algorithms can be difficult to implement.

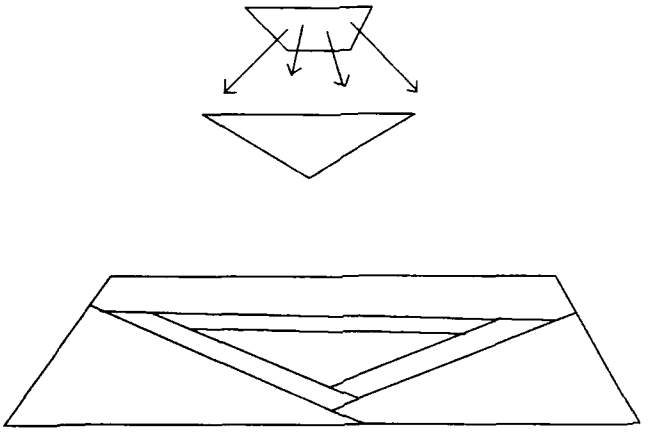

Figure 2.8. Scene efficiently meshed with bsp-tree mesh.

2.5 Extensions

Both ray tracing and radiosity have produced impressive images, but neither method covers all mechanisms of light transport. Attempts have been made to modify each of these techniques to account for the missing terms (28).

2.5.1 The Rendering Equation In 1986, James Kajiya extended ray-tracing, using techniques similar to distributed ray tracing, to more accurately account for diffuse reflection. In a technique known as path tracing, he supersampled each pixel. Instead of always casting a shadow ray (a ray to the light source), a reflected ray and a refracted ray for each intersection, he would always cast the shadow ray. Then using a Monte-Carlo sampling technique, he would cast either a reflective, a refractive or a diffuse ray (Figure 2.9). The direction of the diffuse ray would be randomly determined. By doing this he could get some of the effects of diffuse color bleeding in a ray- tracing approach. (22)

The major problems with this approach are again the computational high cost, and the fact that shading of diffuse surfaces is computed independently for each pixel. Since diffuse surfaces generally change color very slowly, this results in unnecessary work. An-

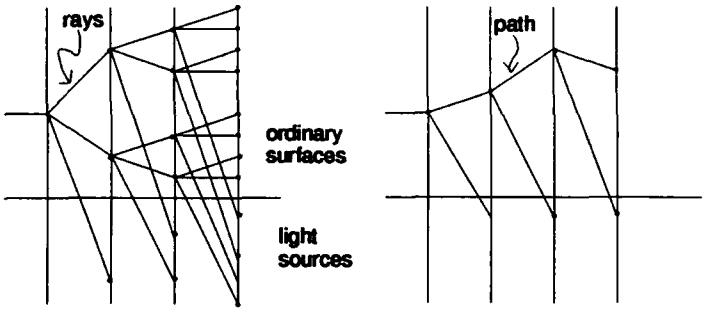

Figure 2.9. (22:148) Diagram of rays shot with traditional ray tracing and the rendering equation method.

other problem is diffuse surfaces that do not get any direct illumination will invariably be under sampled and a satisfactory estimate of illumination will probably not be obtained. (31, 28)

2.5.2 Particle Transport In 1990, James Arvo and David Kirk (1) extended the work done by Kajiya. They realized that the rendering equation as presented by Kajiya is similar to the linearized Boltzman transport equation which has been widely studied in physics and nuclear engineering. Their method involves adapting statistical techniques used in these fields to stochastic ray tracing. The two major additions to previous work involve using *Russian Roulette* to terminate recursive rays without introducing statistical bias, and using a technique known as *Splitting* to make sure areas which would be under sampled with traditional techniques are accurately covered.

2.5.3 Extended Radiosity A way to extend the basic radiosity method to account for specular reflections was introduced in 1986 (21). In this method, the relationship between every outgoing direction for every patch with every outgoing direction for every

other patch had to be determined. This yields a virtually infinite number of unknown intensity values to be simultaneously solved. To make this tractable, not only the surfaces need to be discretized, but also the "hemisphere of incoming and outgoing directions" (21:135). This discretization alone is not enough though, for even the simplest scenes, the number of equations produced becomes intractable. Fortunately, when these equations are presented in matrix form, the matrix is very sparse, usually less than 0.1% full. This fact enables a solution, at least for relatively simple scenes.

2.6 Multi Pass Methods

At this point it becomes clear that neither the ray tracing nor the radiosity method alone are sufficient for creating realistic computer generated images that model all types of reflection. Ray tracing models specular reflections very well, but does not accurately model diffuse reflections. Radiosity handles diffuse reflections well, but ignores the specular. The obvious next step would be to use radiosity to calculate the diffuse and therefore ambient terms, and then use these values with a ray tracer to calculate the specular reflections. (6)

2.6.1 Wallace, Cohen and Greenberg The first two-pass method was developed in 1987 (31). This method uses conventional radiosity to compute the diffuse shading of an environment, then chooses a view and ray traces the scene. During the ray tracing step, whenever a ray lands on a diffuse patch, instead of using the traditional ray tracing approximations, the diffuse intensity is determined by interpolating from the precomputed patch radiosities. Instead of having the overhead of performing an entire radiosity and ray tracing step, the ray tracing step can be reduced. Since shadows have already been calculated with radiosity, shadow rays for diffuse surfaces do not have to be considered. This is a tremendous savings since shadow testing can be the most computationally expensive part of the ray tracing process. Also, to change views, only the ray tracing step needs to be repeated.

While this scheme does produce impressive results, it still does not provide a complete solution to the global illumination problem. As in most other models, the illumination in this model is still computed as the sum of a specular and a diffuse component, as if these terms were completely independent, which they are not. (28, 29)

2.6.2 Sillion and Puech The basic two-pass approach was extended using advanced models that allowed arbitrary reflection modes (28). In this approach more accurate and cost effective ways of calculating form factors are used. In fact, these form factors give the "isotropic" distribution of light, which allow for any number of specular interactions. These new form factors, which take into account both the diffuse reflections and the specular reflections or refractions, are called *extended form factors* and are defined as follows: "F_{ij} is the proportion of energy leaving surface element i and reaching surface element j after any number of specular reflections or refractions /cite[338]SILL89." Sillion and Puech also did away with some of the restrictions of Wallace's method, those being that specular surfaces must be planar mirrors, and that refraction was not handled. This was the first technique that did away with the assumption that specular and diffuse reflections were independent.

2.6.3 Chen, Rushmeier, Miller and Turner Realism in computer graphics was further enhanced again in 1991 (6). Extended progressive radiosity was combined with ray tracing to produce some remarkable results. (Extended progressive radiosity is a progressive radiosity approach that takes specular reflections into account). This approach takes the following three steps to calculate illumination:

1) Progressive Refinement Radiosity (PRR) - uses extended form factors computed by ray tracing for non-diffuse environments.

2) Light Ray Tracing (LRT) - traces rays from light sources (not from the eye) for caustic map generation.

3) Monte Carlo Path Tracing (MCPT) - techniques from particle transport, including *Russian Roulette* and *splitting* (1) are used to determine rays to be cast. Two passes are made with this method, one high frequency to create shadows and caustics, and one low frequency for small details like color bleeding.

The authors claim this approach takes all possible light paths into account. They divide them into four classes. The first of these are direct illumination paths. These are paths from a diffuse-like surface to the eye via zero or more specular-like surfaces. These paths are followed using MCPT. The second class considered is caustic paths. These paths go from at least one specular-like surface to a diffuse-like surface, and then zero or more specular-like surfaces before reaching the eye. These paths are followed using LRT. The third class of light paths are called highlight paths. They consist of zero or more specular-like surfaces and are traced by MCPT. It is these paths that account for the direct rendering of light sources. The final paths considered are termed radiosity paths. These contain at least two diffuse-like surfaces, and are rendered using a combination of PRR and MCPT. These paths produce the classic radiosity effects such as color bleeding.

2.6.4 Sillion, Arvo, Westin, and Greenberg A new approach to solving the global illumination problem was also presented in 1991 (27). This approach extends the progressive radiosity method to include arbitrary reflectance distributions. While this has been done before, (21, 31, 6, 3), all of these approaches relied on a discrete set of directions for calculating the specular terms. This approach uses continuous functions to encode the directional dependence of intensity distributions and thus introduces no such restrictions.

This method uses the reflectance classification introduced by He and Torrance (19) which divides reflections into ideal diffuse, ideal specular and directional diffuse. Standard radiosity and ray tracing handle the first two cases; it is the directional diffuse light that needs special consideration. The radiosity step here is again extended to handle the directional nature of reflections, this time using a bidirectional reflectance distribution function

(BRDF) approximated using spherical harmonics to store the intensity distributions. This allows for tremendous storage savings and increased accuracy over the traditional hemi-cube methods.

In this two-pass algorithm, an extended progressive radiosity method is used to calculate a view-independent solution for the directional diffuse distribution of light, which includes the effects of intermediate specular reflections. The basic shooting algorithm now uses the directional intensity distribution emitted by the shooting patch to update the directional intensity distributions of all the receiving patches. The second pass uses ray tracing to compute the view-dependent ideal specular effects, exactly what ray tracing does best. The difference between this and a standard ray tracing algorithm is that this method uses a specular reflectance function to accurately model the specular reflections, instead of the traditional specular coefficient used in ray tracing. This approach differs from previous two-pass methods in that it now accounts for all possible light paths and incorporates arbitrary reflectance distributions. (27)

2.7 Conclusion

A brief history of work in the area of realistic image synthesis has been presented in this chapter. Through the years the degree of realism in computer graphics has greatly increased, from the early work where just getting some kind of a specular highlight was a major accomplishment (26), to the point now where researchers are claiming to model all possible light paths (6, 27). Despite these accomplishments, much remains to be done in this area. One area of further research is to continuously increase the level of realism. Another area is to decrease the computation times of the existing techniques.

III. Diffuse Radiosity Design

3.1 Introduction

This chapter describes the design of the diffuse radiosity based realistic image generation system. Included is a discussion of the design of each of the major components, with emphasis placed on design decisions made during the development process.

3.2 Starting Point: Basic Progressive Refinement Radiosity

To get started, a basic radiosity capability was needed. There were three primary choices: use radiosity code developed by previous CSCE 682 students, use code provided with the book *Graphics Gems II* (2), or develop a new program.

Initially, the code developed by previous CSCE 682 students was chosen because of its ready availability. It was known that the code did not work properly, but it was assumed to be an easy fix. It soon became apparent that this code was not well written and would not be worth pursuing.

Of the remaining two choices, developing a new program would have required too much time and not allowed sufficient time to look at the more advanced features that this research was to investigate. This left the *Graphics Gems II* code as the only viable option.

The radiosity code provided with *Graphics Gems II* was written by Shenchang Eric Chen, and was designed to be a simple and straightforward progressive radiosity implementation. It takes as input a polygonal description which is hard-coded into one of the routines and breaks it up into patches and elements in a pre-defined manner. Form factors are calculated with the hemi-cube method utilizing hardware z-buffer capabilities of the host workstation. The user is responsible for writing all of the drawing routines so the provided software can be machine independent.

The hardware z-buffer is utilized by opening a window with the same resolution as the hemi-cube, and setting up a view for each side of the hemi-cube. The entire scene is then rendered into this window for each side of the hemi-cube using the element number for the color value. The user provides a routine that will read this information from the screen and into a buffer. Each pixel value has pre-defined delta form factors that are summed to find the form factors between the shooting patch and a given element.

The basic data structure for this implementation is an array of elements which point to their corresponding patches, and an array of patches. This allows easy access to each element or patch in the array as long as the number of elements and patches is known in advance and remains static.

This implementation has a number of limitations. Flexibility is limited, since the polygonal description of the room is hard-coded. All of the elements are flat shaded, and the data representation does not make it easy to determine element adjacency on a given polygon to do interpolation for more realistic shading. A more subtle limitation is the fact that the patches and elements need to be pre-defined. This results in difficulty in knowing how to break the polygons up; dividing a scene up too fine wastes time, but if the division is too course, important details like shadows are missed.

3.2.1 Capabilities This system shows the basics of a progressive refinement radiosity program. It only shows the Cornell University room (Figure 3.1). No capability is given to change the image resolution or the way the polygons are broken up. All elements are flat shaded, and the image can only be seen while the program is running; as soon as the program finishes, the image disappears.

3.2.2 Comments This software gave a basic capability from which to start development. It was logically laid out and modular. Before development was complete, most of the modules were completely replaced and the rest were changed significantly.

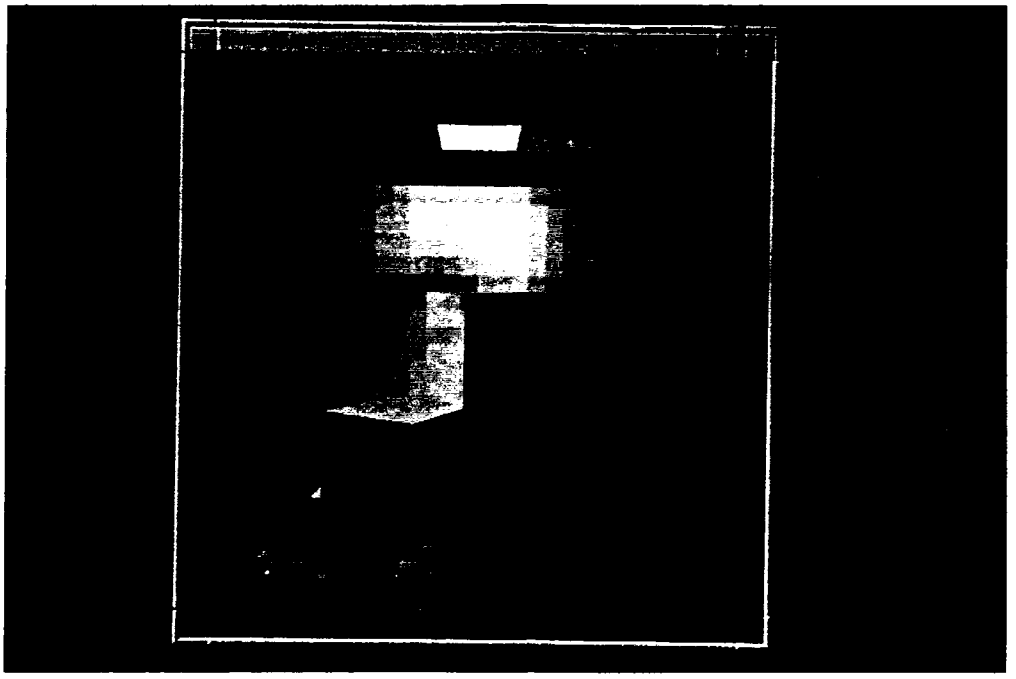

Figure 3.1. Original Cornell Room

3.3 Automated Meshing

One of the greatest challenges in developing a radiosity program is the meshing of the environment. There are many factors to consider when choosing a meshing algorithm, including algorithm complexity and radiosity performance. Complexity is important because of the amount of time it takes to develop good automated meshing software, and this is especially true when developing a good meshing algorithm is not the primary goal of the research. Radiosity performance is an important consideration when designing a meshing algorithm because the more elements an environment has, the longer it takes to produce an image, but if the environment are not divided finely enough, the final image will not be accurate. With these factors in mind it is necessary to design an algorithm that will allow fine meshing in areas the radiosity is varying quickly and larger meshing in areas of relatively constant radiosity. It also must be fairly easy to implement, since developing a meshing algorithm is not the primary focus of this research.

It was decided to allow the user to specify how each polygon would be broken into patches, but to let the program have control over how the patches are broken into elements.

The patch subdivision algorithm works as follows: Shoot from patch with highest radiosity, and at each vertex keep a value that tells whether or not it was visible to the shooting patch. Then, go through each edge of each element, if the edge is visible at one vertex and not visible at the other, this edge lies on a shadow boundary, otherwise, it is not. This determination is made because patches at shadow boundaries are treated differently from other patches (A different threshold is used for the allowed radiosity difference across that edge). Compare the radiosities at those vertices. If the difference is over some threshold, and the area of the element is over some predefined minimum, divide that element. The next step is to shoot from the shoot patch to all of the newly created vertices. Continue applying this algorithm recursively until all edges fall within thresholds. To account for the effect of all of the previous shooting patches on these newly created vertices, something else needs to be done. When new vertices are created, the accumulated radiosity for the new vertex is interpolated from the vertices on either side of it. This interpolation is accurate because only the effects of previous light sources are interpolated. If the difference in radiosity between the two interpolating vertices is so large that a linear interpolation would not be accurate, the element would have already been subdivided.

3.4 Data Structures

To allow the elements to be created at run time, a data structure that allows dynamic allocation is needed. When elements and patches were both statically determined, arrays were acceptable, but since the program should determine which areas need more meshing, a different data representation is required. It is desirable to only store information for each vertex once.

The data representation chosen uses an array of patches, each pointing to a quad-tree

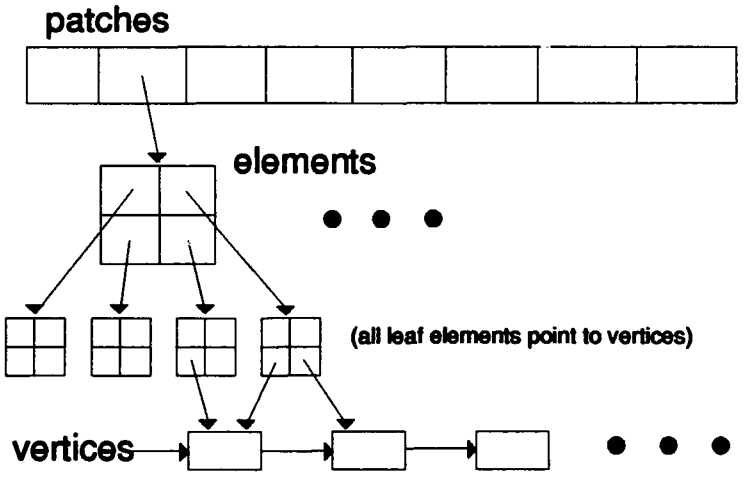

Figure 3.2. Data structure used for patches, elements, and vertices

of elements. Each element then points to the four sub-elements and also to its four vertices. The vertices are stored once each in a linked list. This entire structure is illustrated in Figure 3.2. With this structure, all of the elements can be accessed by looping through the patches and doing a recursive traversal of the elements under each patch. The vertices can be accessed independently by traversing through the linked list. At program initialization, before any radiosity has been shot, the data is simply an array of patches. Each of these patches contains a pointer to an element that stores the patch's geometric information. The element trees all have a depth of one at initialization. A quad-tree data structure was chosen because it allows for easy division of patches and is easy to traverse.

The automated meshing works on this data structure as follows: When it is determined that an element edge needs divided, that element is divided into four elements that are pointed to by the original element. It is the leaf elements that are of interest in calculating patch radiosities and rendering the image.

A major concern with this implementation is the introduction of T-vertices when using the quad-tree. As the quad-tree is subdivided in a nonuniform manner, more and

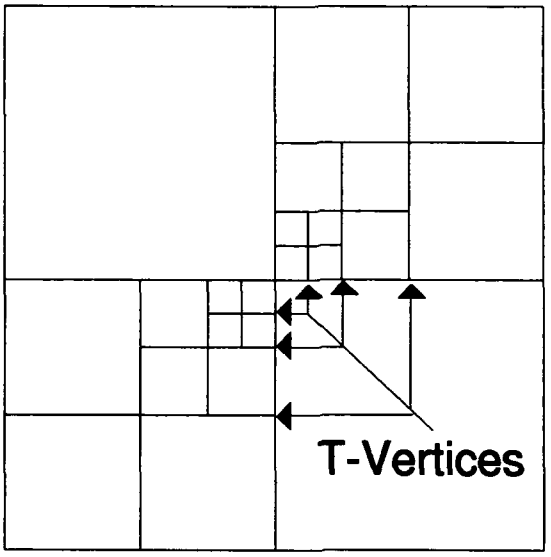

Figure 3.3. T-Vertices

more T-vertices appear (Figure 3.3) and these cause two problems. The first is a shading problem. Gouraud shading interpolates intensities along edges of each polygon. Therefore, when one polygon is adjoined on a single edge by two, four, or even more polygons, the interpolated values along the edge of the single polygon may not match the calculated values of the neighboring points on the adjoining polygons. This can cause some obvious visual artifacts. The second problem stems from the fact that the new points are calculated using floating point arithmetic and the display is mapped by integers. The calculated point may lay on the edge adjoining the larger polygon in world space, but when it is mapped to the screen, it may be one pixel off, resulting in a row of pixels that do not get colored (Figure 3.4). This problem is termed pixel-drop.

The problem with the Gouraud interpolation is corrected by the nature of the adaptive subdivision. Anywhere that a linear interpolation is not very accurate is subdivided further, so problems of the first type mentioned above are minimal. The second problem however can be quite obvious. Even one case of pixel drop can ruin an otherwise good image. To combat this, the new vertices created by splitting a quad-tree element are calculated in double precision. This method is adequate to insure no pixel-drop occurs.

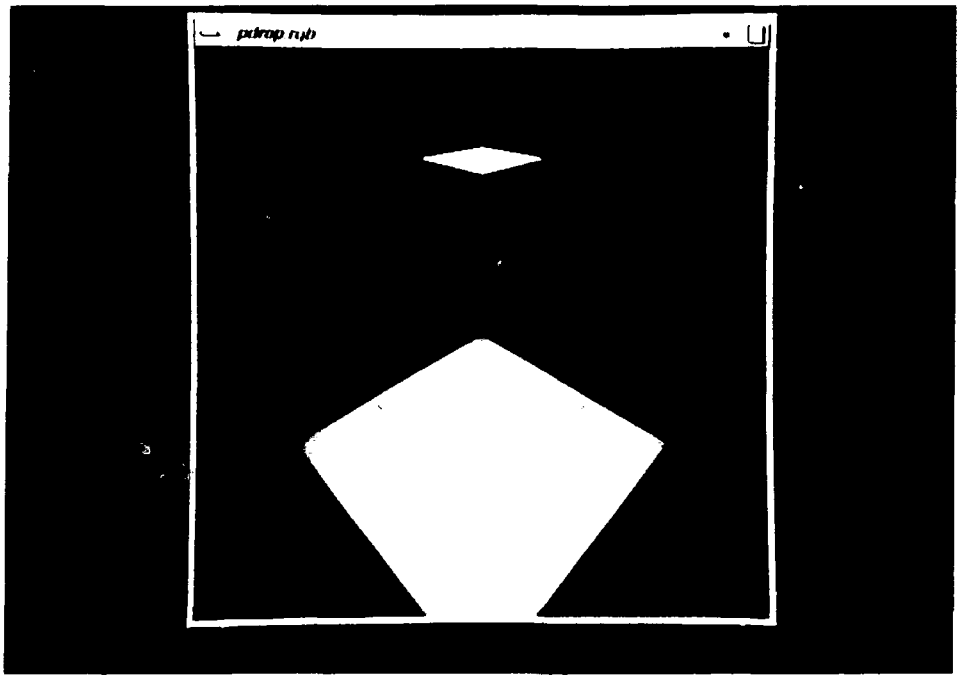

Figure 3.4. Image showing black dots where pixel drop occurred

It is also necessary to keep track of the original input polygons. Otherwise the only polygons available for ray-polygon intersection tests are the elements. Since there can be hundreds of elements per polygon, this is an unacceptable solution. This implementation uses a simple array of polygons which index into an array of points and an array of normal vectors to represent the input polygons.

3.5 Input Files

Since the radiosity code used as a basis for this research had no capability to read in a geometry file for the polygonal description of the environment, this capability had to be developed. Because of the wide availability of objects already modelled in the AFIT geometry file format, a modified version of this format was chosen. The modification was necessary to allow for the specification of emittance for light sources and to allow the user to specify how far the polygons should be broken up into patches. The format for this file is given in Appendix B.

It was also desirable to put all of the viewing parameters into an input file. This file specifies such attributes as eye and reference locations, field of view, up vector, and image resolution. Format for this file is given in Appendix A.

3.6 Ray Traced Form Factors

Moving from using the hemi-cube to calculate form factors to using ray tracing was necessary for a couple of reasons. The primary reason is the fact that the ray traced form factors are required for the extended radiosity implementation that was the primary goal of this research. The other reason is the advantage of calculating the radiosity at the vertices instead of the center of each element. This decreases the aliasing problems experienced with the traditional hemi-cube method.

The technique chosen for calculating the ray traced form factors was introduced at SIGGRAPH '89 in a paper entitled "A Ray Tracing Algorithm for Progressive Radiosity" (32). This technique differs from previous works with ray traced form factors in that all the previous methods traced rays from the shooting patch in some sampling of directions over the hemisphere to determine the form factors (25). The result of doing it in this manner is a patch radiosity that must be interpolated to the vertices. This creates the additional burden of keeping track of which elements border each of the others. Another problem with this method of tracing rays is the fact that the knowledge of where the other surfaces are is not being used, therefore, some areas are being over-sampled, while others are not being adequately sampled.

The chosen technique traces rays from each vertex to points on the shooting patch, so the radiosity is calculated for each vertex. This technique also allows varying the sampling of the shooting patch, giving a good approximation with a low number of rays and increasingly accurate results with higher numbers of rays. These additional rays are required for accurate modelling of the penumbra at the shadow boundaries.

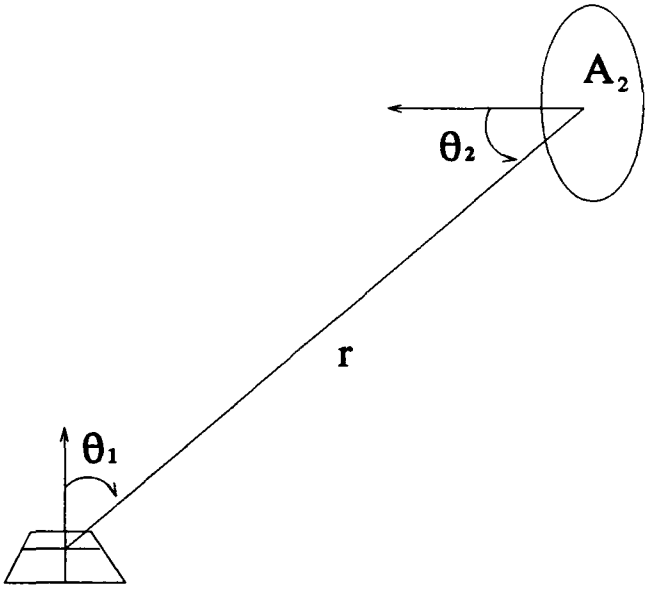

Figure 3.5. Arrangement of θ_1, θ_2, and r

The basic equation for the form factors in this method is:

$$dF_{1,2} = dA_1 \frac{1}{n} \sum_{i=1}^{n} \delta_i \frac{\cos\theta_{1i} \cos\theta_{2i}}{\pi {r_i}^2 + A_2/n} \tag{3.1}$$

where n is the number of samples taken of the shooting patch, r, θ_1 and θ_2 are as shown in figure 3.5, and δ_i is the visibility factor. This factor is 1 if the vertex is visible to the receiving patch and 0 otherwise. For a complete derivation of this form factor equation see (32:318-319). For this implementation, n was chosen to be one.

3.7 Explanation of Algorithm

The psuedocode for the radiosity process used in this research is given in Figure 3.6. The entire radiosity process is now as follows:

1. Read in the input geometry file. The result of this is three arrays. One with vertex points, one with vertex normals, and one with information about the input polygons with

```
Read Geometry File
Read Display File
Initialize Patches and Vertex List
Loop until converged
{
        Find patch with most Unshot radiosity (SP)
        Shoot(SP)
        If SP is light source
                AutoMesh(SP)
        Display Image
}
Save Image

Function Shoot(SP)
{
        For each Vertex (V)
        {
                If V can see SP
                {
                        Calculate Form Factor (F)
                        ΔRad = SP.unshotRad * V.reflectance *F
                        Add ΔRad to total and unshot radiosity for V
                }
        }
}

Funtion AutoMesh(SP)
{       For each Element
        {
                if (Radiosity difference across Element > threshold)
                        Divide Element
        }
        Shoot from SP to new vertices
        AutoMesh(SP)
}
```

Figure 3.6. Psuedocode for radiosity program

indices into the other two arrays.

2. Read in the display attributes file.

3. Initialize the patches. Initialize head and tail pointers for the linked list of vertices. Allocate space for the array of patches. Divide the input polygons into patches. To only have each point represented once, the arrays of vertices and normals are converted to a linked list of vertices with an array of pointers pointing to them. This allows assignment of vertex pointers in the same manner as indices into the vertex array were assigned in the *Graphics Gems* implementation.

4. Initialize the Radiosity Parameters. This passes all of the necessary variables that are global to the radiosity calculating routines. It also calculates the total energy in the scene which is used to determine if an adequate solution has been reached.

5. Main Radiosity Loop. The following steps are repeated until an acceptable solution has been reached.

5a. Find the patch with the most unshot radiosity. This requires going through the array of patches. At each patch a recursive traversal of all of the elements under that patch is made, taking an area weighted average of all of the unshot radiosities of the vertices within the patch. This is accomplished by using a weighting factor that divides the radiosity by four at each level down the tree. This area weighted average unshot radiosity is then added to the unshot radiosity of the patch and the vertex unshot radiosities are returned to 0. When this is done, the patch which now has the highest unshot radiosity is selected as the new shooting patch. If the ratio of the unshot energy in the shooting patch to the total energy in the scene is below a threshold, the scene is adequately rendered and the process stops.

5b. Shoot radiosity from shooting patch to all vertices. This is done by going through the linked list of vertices and tracing rays from each of the vertices to the shooting patch. If a polygon intersects the ray, the form factor is 0, and that patch has no effect on the

radiosity at that vertex. If no polygon intersects the ray, the radiosity at that vertex due to the shooting patch is $\rho_1 B_2 F_{21}$ where ρ_1 is the reflectivity of the vertex, B_2 is the unshot radiosity of the shooting patch, and F_{21} is the form factor between the shooting patch and that vertex. Both the vertex's total radiosity and unshot radiosity are incremented by this amount.

5c. If the shooting patch was one of the emitters (a light source), the vertices within elements are examined to see if the element is on a shadow boundary and needs to be broken down finer to accurately represent that shadow. This is done in the same recursive manner as finding the shooting patch. If an element is found that needs to be broken down finer, that element is divided into four elements. The radiosity from the shooting patch to each of the newly created vertices is then calculated. This is done by shooting radiosity from the shooting patch to the new vertices. The division criteria is again applied recursively.

5d. Display the results of that pass. Again a recursive traversal through the elements is made, this time element vertices and radiosities are extracted and polygons are drawn. This allows the image to be seen as it is being generated.

6. Save the picture to a file for later viewing. This is done by saving for each polygon: four points and their corresponding intensity distribution. For the diffuse case, the intensity distribution is simply the r, g, and b color values.

3.8 Viewing the Image

Since the color values for all of the vertices are stored in a file, this information can be used by an interactive viewer to see the image. This is one of the best features of a radiosity implementation. Even though it can take a long time to calculate the radiosity for an image, the results can be displayed in an interactive manner because of the view-independence of the resulting image.

The viewer used for this research has the following capabilities:

- Looking in any direction.
- Zooming in or out.
- Moving forward or backward.

These same features are also available using a head mounted display (HMD).

3.9 Conclusion

All of the design decisions made for the diffuse radiosity system represent trade-offs. Since the primary goal of this research was to implement an extended radiosity system that incorporated specular effects, design decisions were made that most readily led to that goal. Using ray traced form factors to the vertices is the foundation of this design and allows for an easy transition to more complex extended radiosity implementations.

IV. Directional Diffuse Radiosity Design

4.1 Introduction

This chapter describes the design of the extended radiosity system. The extensions involve taking specular effects into account when calculating the radiosity of an image. The focus is on areas where the design is different than that of the diffuse implementation.

4.2 Background and Overview

Traditionally, radiosity has only dealt with diffuse reflections. While this can give some very pleasing results, and does model the diffuse portion of the global illumination in a scene, it still neglects a large portion of the visual effects one expects to see. Various approaches to extending the radiosity method to include specular effects have been attempted and are discussed in chapter 2. Most of these have shortcomings in terms of performance or accuracy. The method chosen for this work was introduced at SIGGRAPH '91 in a paper titled, "A Global Illumination Solution for General Reflectance Distributions" (27). This method differs from the others primarily in the way it keeps track of the reflectance of the surfaces. While all of the other approaches use a discretized representation, this approach uses continuous functions at each vertex to represent the reflectance and store the intensities. This continuous approach has two primary advantages over the discrete approach. The first of these is the aliasing problems inherent in any discretizing of a continuous function. The second problem is the large amount of storage required to save all of the discrete values required to minimize the aliasing problems. These reflective properties are described with a ***bidirectional reflectance distribution function*** (BRDF), "which is defined as the ratio of the reflected radiance in a given outgoing direction to the incoming energy flux (per unit area) in another direction (27:187)."

Overall, this implementation is the same as a standard progressive radiosity implementation except for the fact that continuous directional distributions are used in place of

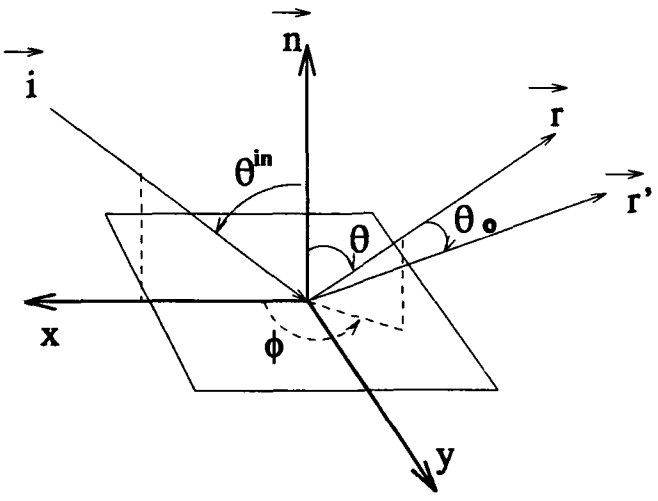

Figure 4.1. Angles and vectors used in reflectance equation. Light comes in along direction $\vec{i}$ and leaves in direction $\vec{r'}$. $\vec{r}$ is the direction of an ideal reflection. θ^{in} is the angle between $\vec{i}$ and the surface normal (n). θ is the angle between $\vec{r'}$ and n. θ_o is the angle between $\vec{r}$ and $\vec{r'}$. ϕ is the azimuthal angle between x and $\vec{r'}$.

diffuse radiosities. "The basic *shooting* operation now uses directional intensity distribution emitted by the shooting patch to update the directional intensity distributions of the receiving vertices... (27:189)".

4.3 Representing BRDFs

For this research, BRDFs are represented using a truncated Legendre Polynomial expansion. This is possible because the BRDF ρ, is dependent only on the angle θ_o (see Figure 4.1). This allows for a very compact representation of the BRDFs.

$$\rho \propto e^{\left[\frac{\tan^2(\theta_o/2)}{\alpha^2}\right]} \tag{4.1}$$

where α is the standard deviation (RMS) of the surface slope. Let $\mu_o = \cos\theta_o$ and simplify:

$$\rho \propto e^{\left[\frac{-(1-\mu_o)}{\alpha^2(1+\mu_o)}\right]} \tag{4.2}$$

Expand in a truncated Legendre polynomial series:

$$\rho(\mu_o) \approx c_R \sum_{l=0}^{L} a_l P_l(\mu_o) \tag{4.3}$$

where

$$a_l = \int_{-1}^{1} \frac{2l+1}{2} P_l(\mu_o) \exp\left[\frac{-(1-\mu_o)}{\alpha^2(1+\mu_o)}\right] d\mu_o \tag{4.4}$$

and c_R is a normalization constant, chosen to assure the total reflection fraction is K_s (the specular constant for the surface). L is chosen based on the α value for the surface, a small α requires a larger L. This is because it takes more spherical harmonic expansion terms to accurately represent functions that are more specular. A full derivation of the BRDF representation is found in Appendix G.

With this representation, the only things that needs to be stored to represent the BRDF at each vertex are the a_l terms. The actual reflectivity of a surface is calculated given the particular geometry of the instance at the time it is needed.

Because the calculation of the a_l terms requires an integral, and these only need to be calculated once per surface type, it was decided to separate this calculation from the rest of the program. All of the coefficients are calculated in advance with another program and saved in a data file. Format for this file is shown in Appendix F. This only needs to be done once for each surface type. There are a number of different ways these terms can be calculated. The easiest of these to implement was to use a *Mathematica* program (36) to find the a_l terms. A listing of the *Mathematica* program used for these calculations is in Appendix E.

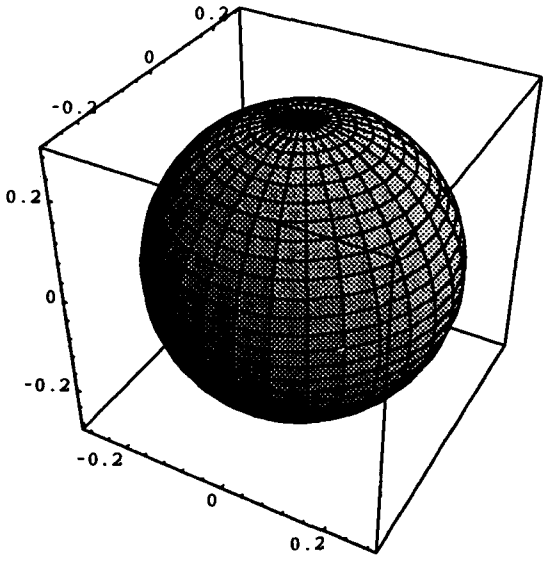

Figure 4.2. Plot of $|ReY_{0,0}|$.

4.3.1 Data Structures for BRDFs The only thing that needs to be stored for representing the BRDFs is the a_l terms. For this reason the data structure used is a dynamically allocated array of $L + 1$ values, where L is the largest l value in the spherical harmonic expansion for that surface. This array is dynamically allocated because L can range from zero for a completely diffuse surface up to 25 for a highly specular surface.

4.4 Representing Intensities

The approach chosen to represent arbitrary intensities uses *spherical harmonics* as described in (27). "Spherical Harmonics form an orthogonal basis for the space of functions defined over the unit sphere. This infinite collection of basis functions is typically denoted by $Y_{l,m}(\theta, \phi)$ where $0 <= l <= \infty$ and $-l <= m <= l$ (27:192)." The three dimensional surfaces that represent the BRDFs can be described with weighted sums of these basis functions. Plots of different $Y_{l,m}$ terms are shown in Figures 4.2 through 4.5. Figure 4.2 shows the $Y_{0,0}$ function is constant in all directions, this is used to represent the diffuse terms. For more information on spherical harmonics see (27) or (13).

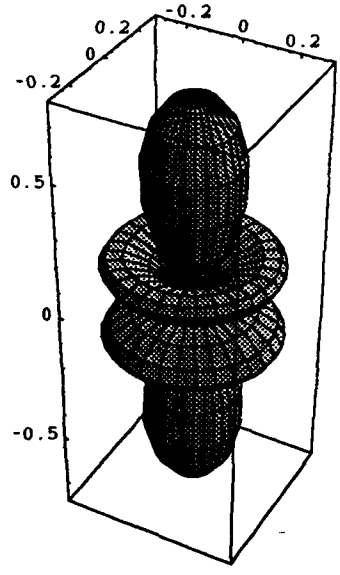

Figure 4.3. Plot of $|ReY_{3,0}|$.

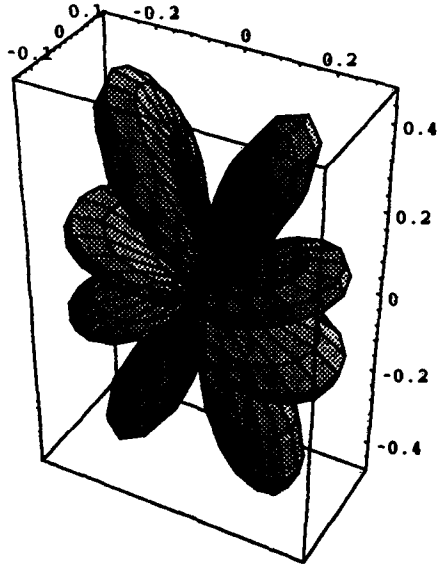

Figure 4.4. Plot of $|ReY_{4,1}|$.

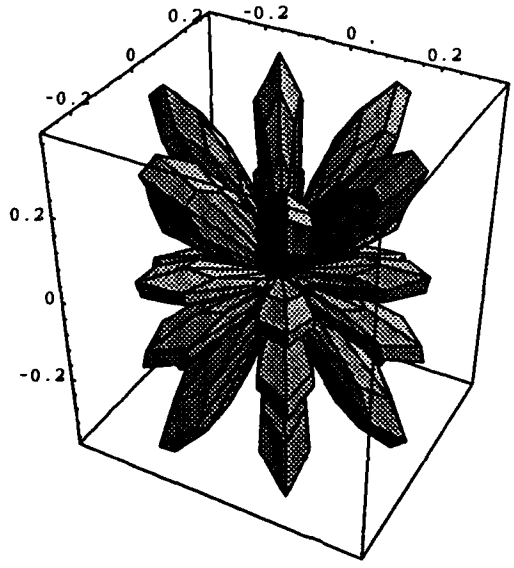

Figure 4.5. Plot of $|ReY_{6,3}|$.

The equation for intensity is:

$$I(\cos\theta, \phi) \approx \sum_{l=0}^{L} \sum_{m=-l}^{l} B_{l,m} P_l^m(\cos\theta) \cos(m\phi). \tag{4.5}$$

$B_{l,m}$ is simply a scalar value. With the intensity distributions, both the positive and negative m terms need to be stored because there is no symmetry with respect to the incident plane when the intensities from different light sources are combined.

4.4.1 Data Structures for Intensities To limit the amount of storage required to represent arbitrary intensity distributions, the only part of equation 4.5 that has to be stored is the $B_{l,m}$ terms. Since the number of terms required to represent the BRDFs is not known until run time, the data structures used must be dynamic.

The data structure chosen for this implementation used a dynamically allocated array of pointers to dynamically allocated arrays. Because the array is dynamic in both dimensions, an array of l pointers is allocated. Then for each of the i elements of this array ($0 < i < l$), an array of $2i + 1$ elements is allocated (to store all m terms ranging from -i

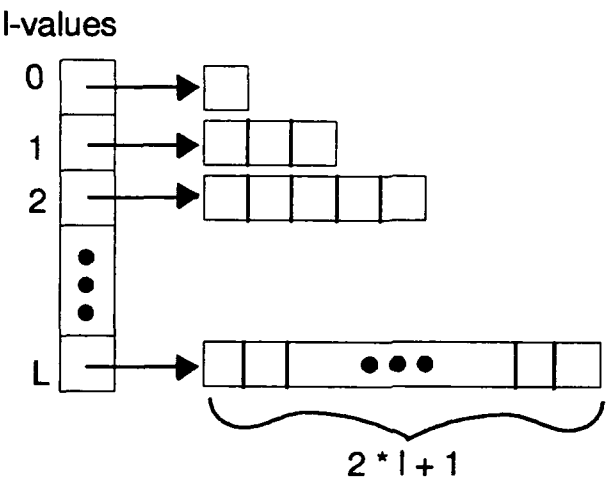

Figure 4.6. Data structure used for representing intensities.

to i). A pictorial representation of this data structure is shown in Figure 4.6.

4.5 Calculating Radiosity Distributions

The basic algorithm for calculating the extended radiosity is essentially the same as that for diffuse radiosity given in the previous chapter. The patch with the most unshot radiosity is still the shooting patch, and the shooting patch distributes radiosity to all of the vertices. The primary difference is that the outgoing and incoming directions of the radiosity are now significant.

4.5.1 Explanation of the Algorithm The psuedocode for the extended radiosity implementation is shown in Figure 4.7. As the geometry file is being read in (just as in the diffuse radiosity case in chapter 3), the BRDFs for each surface are built up as described in the previous section. The input file needs only constants for diffuse and specular reflectivity, which are already part of the *AFIT geom* file format, and an alpha value, which is the standard deviation (RMS) of the surface slope.

For this implementation, light sources were considered diffuse. This made building up

```
Build up BRDF for each vertex
Build up Intensity distribution for each light source
Loop until converged
{
        Find patch with most Unshot radiosity (SP)
        Shoot(SP)
}

Function Shoot(SP)
{
        For each Vertex (V)
        {
                If V can see SP
                {
                        Calculate Form Factor (F)
                        Find Intensity (I) leaving SP toward V
                        Calculate Flux (Φ) = F * I
                        Find Incident angles in local axis at V
                        Retrieve Reflectivity (ρ) of V for incident angles
                        Multiply ρ * Φ to get change in intensity (ΔI)
                        Rotate ΔI in local axis according to incident direction
                        Add ΔI to total and unshot radiosity distributions
                }
        }
}
```

Figure 4.7. Psuedocode for extended radiosity implementation.

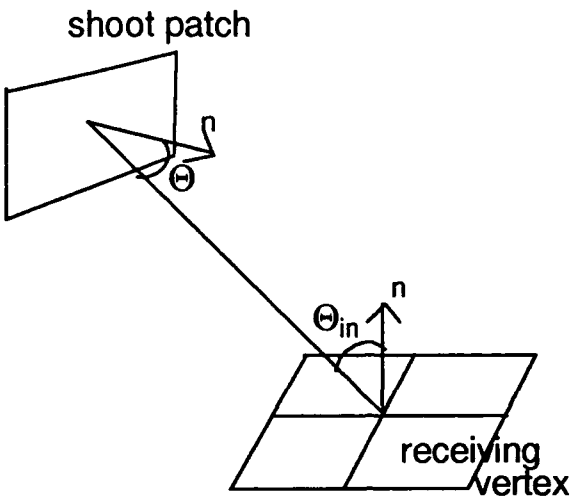

Figure 4.8. Angles used for intensity calculations.

the intensity distribution for the light sources a simple procedure. All that was required was to assign the emit values from the geometry file to the [0,0] term of the spherical harmonic expansion.

Finding the patch with the highest intensity to be the shooting patch was accomplished by using the same recursive traversal of the tree of elements that was used with the diffuse implementation. In this implementation only the [0,0] terms were considered in determining the brightest patch. This is equivalent to finding the patch with the most diffuse light.

Determining if vertex V can be illuminated by the shooting patch is done by the same process as before. It is simply done by casting a ray from the vertex to the shooting patch and seeing if that ray intersects any of the input polygons. If it does not, the vertex is visible from the shooting patch. The calculation of the form factor is also the same as with the diffuse implementation.

To calculate the unshot radiosity (intensity) leaving the shooting patch toward the receiving vertex, a couple of angles must be calculated. These are ϕ and θ. θ is shown in

Figure 4.8 and is calculated by taking the dot product of the shooting patch normal and the ray going from the shooting patch to the receiving vertex. ϕ is calculated by taking the dot product between the ray and the X- axis. Intensity is then calculated by equation 4.5, where the $B_{l,m}$ terms are the unshot radiosity coefficients for the shooting patch. The flux (Φ) is then calculated by multiplying this intensity value by the form factor.

Since the intensities are stored in a local axis at each vertex, the incident angles for the incoming radiosity must be calculated in terms of these local axes. θ_{in} is the angle between the vertex normal and the incoming ray as shown in figure 4.8. This angle is calculated by taking the dot product of the two rays. $\Delta\phi$ is the angle between the incoming ray and a pre-defined basis X-axis for that vertex.

Retrieving the reflectivity for the given incident angles involves building up the BRDF from the stored a_l terms for the given angles ($\theta^{in}, \theta, and\ \phi$). This is done in the following steps:

1. Generate a table of $P_l(\cos\theta^{in})$ values for l ranging from 0 to L.
2. Compute $b_{l,0}$ terms as though $C_R = 1$: $b_{l,0} = a_l P_l(\cos\theta^{in})$ for l ranging from 0 to L.
3. Compute normalization: $C_R = K_s/(2\pi \sum_{l=0}^{L} b_{l,0} C_l)$
4. Normalize $b_{l,0}$ terms: $b_{l,0} = b_{l,0} C_R$ for l ranging from 0 to L.
5. Include diffuse component: $b_{0,0} = b_{0,0} + K_d/2\pi$
6. Compute other b terms: $b_{l,m} = C_R * a_l * d_{l,m} * P_l^m(\cos\theta^{in})$ for l ranging from 0 to L, and m ranging from 1 to l. $d_{l,m} = \frac{2(l-m)!}{(l+m)!}$

These values are multiplied by the flux (Φ) to get the change in intensity (ΔI).

Before this ΔI can be added to the accumulated and unshot radiosities at the vertex, it must be rotated to line up with the local axis of the vertex. This is the point where all of the negative m terms get filled in with the $B_{l,m}$ array. The positive m terms are simply

$\Delta I_m * \cos(m\phi)$ and the negative m terms are $\Delta I_m * \sin(m\phi)$. Now the ΔI terms are added to the accumulated and unshot radiosity for the vertex.

4.6 Viewing the Image

Viewing the image requires calculating the angles between each vertex and the eye of the viewer in local coordinates for that vertex. These are used to evaluate the intensity for that vertex in the given direction. These values are then provided to a renderer and Gouraud shaded as before. If the viewing position is changed, all the intensity values must be re-evaluated.

4.7 Conclusion

The design of the extended radiosity system goes beyond a standard diffuse radiosity approach and allows specular surfaces to be modelled. This approach uses continuous functions to represent the radiosity at each vertex in the environment. Spherical harmonic decomposition is used to represent these functions. A major departure from other implementations of extended radiosity is the way the BRDFs are represented. Instead of having to save $b^j_{l,m}$ terms where 0 <= l <= L and 0 <= m <= l and j's represent discrete increments in θ^{in} as in (27), now only a_l terms need to be saved. This not only results in a large storage savings, but also saves having to interpolate for discrete θ^{in} samples. æ

V. Results

Due to the fact that this research effort produced two products, a diffuse radiosity implementation and one that adds specular effects resulting in an extended radiosity implementation, this chapter is divided into two major sections, one for each product.

5.1 Progressive Radiosity

The program *prorad* is a full featured progressive radiosity rendering program. It allows a user to specify an input geometry in a slightly modified *AFIT geom* format (see Appendix B), and it produces a file containing a diffuse radiosity description. Instructions for using *prorad* are found in Appendix D. This image can then be viewed on the computer screen or in a head mounted display (HMD). Either viewing method allows the user to interactively move around the image. The image is also displayed as the rendering proceeds, with more information showing up at each pass.

5.1.1 Automatic Meshing The program starts running with the initial set of patches as specified by the user. The effect of each light source is then taken into consideration when determining how finely to divide up the patches into elements. The radiosity gradient across each element is considered to see whether it should be subdivided. This is repeated up to a minimum area for each element. An example of a scene in various stages of division is shown in Figures 5.1 through 5.10. The first is the Cornell room showing patches as specified by the user. The remaining figures show the same scene after increasing levels of automatic subdivision.

5.1.2 Progressive Refinement Progressive refinement radiosity in which the brightest patches are always chosen to be the shoot patch converges toward the solution very quickly compared with the standard radiosity approach. This research effort took advantage of that fact by searching through all of the patches and finding the one with the

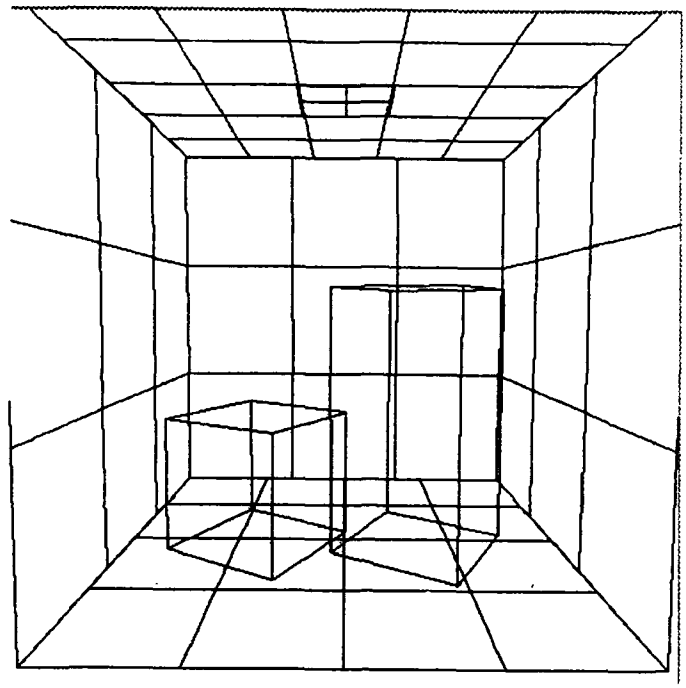

Figure 5.1. Cornell room showing initial patches.

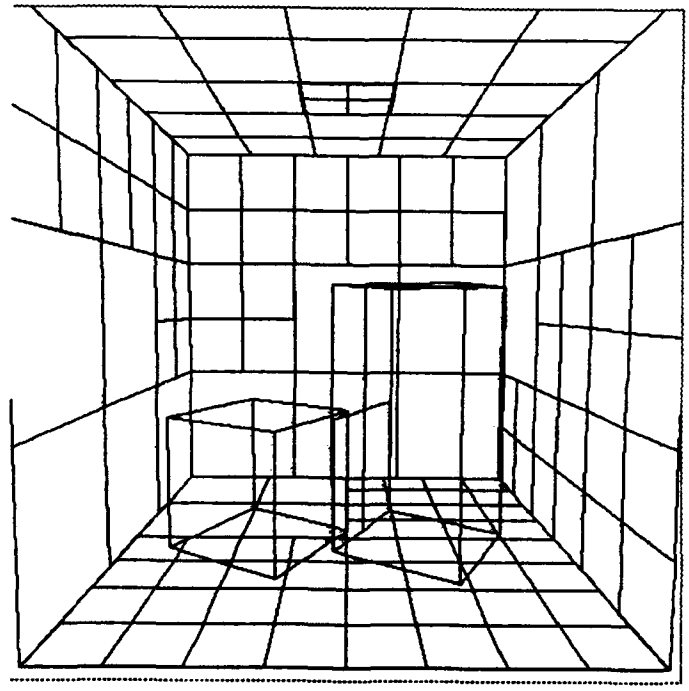

Figure 5.2. Cornell room showing subdivision after 1 level.

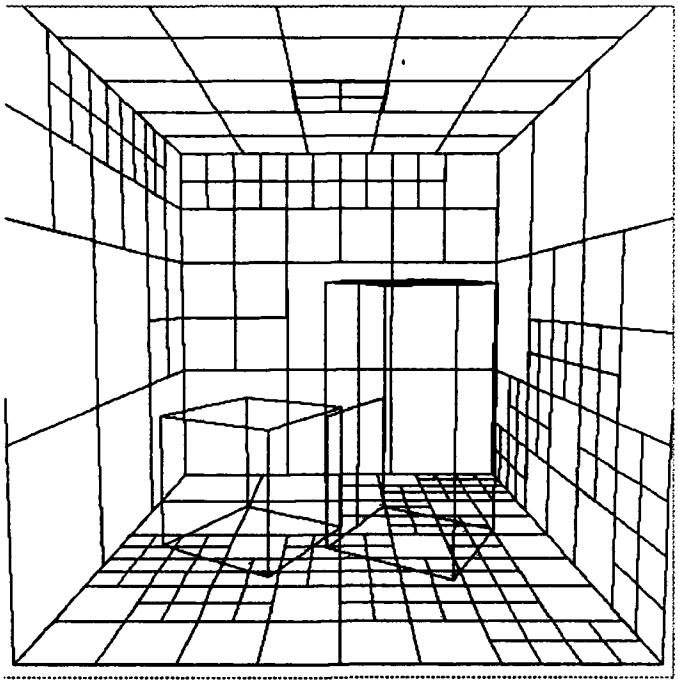

Figure 5.3. Cornell room showing subdivision after 2 levels.

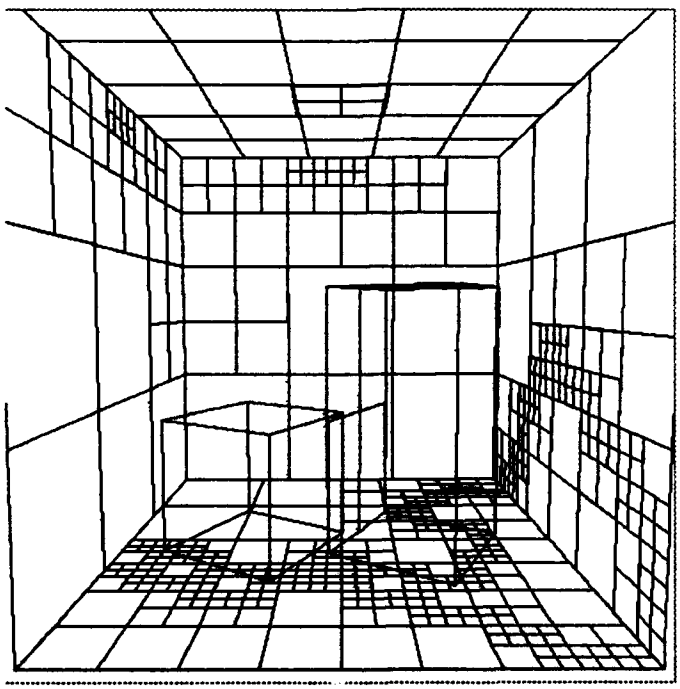

Figure 5.4. Cornell room showing subdivision after 3 levels.

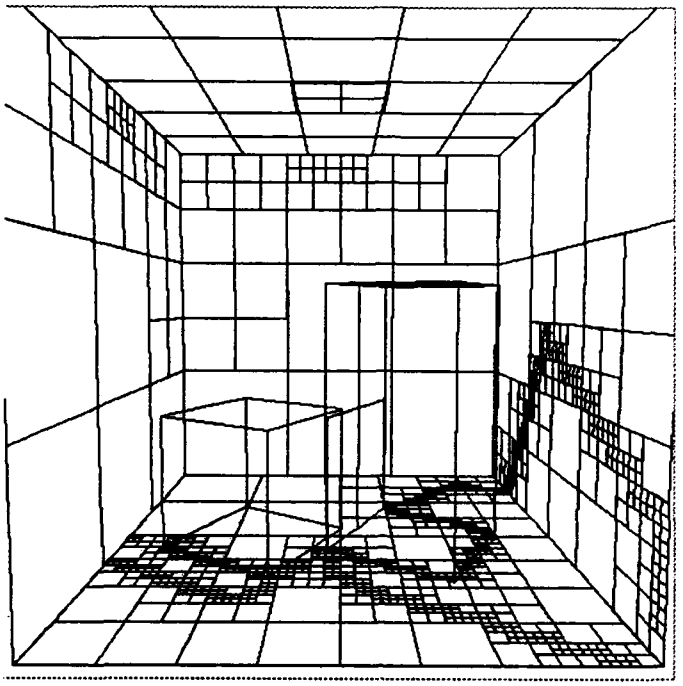

Figure 5.5. Cornell room showing subdivision after 4 levels.

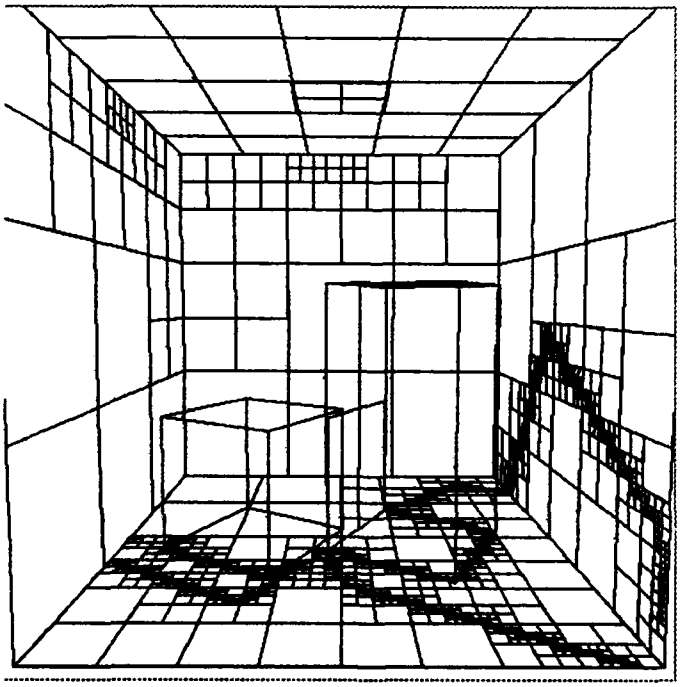

Figure 5.6. Cornell room showing subdivision after 5 levels.

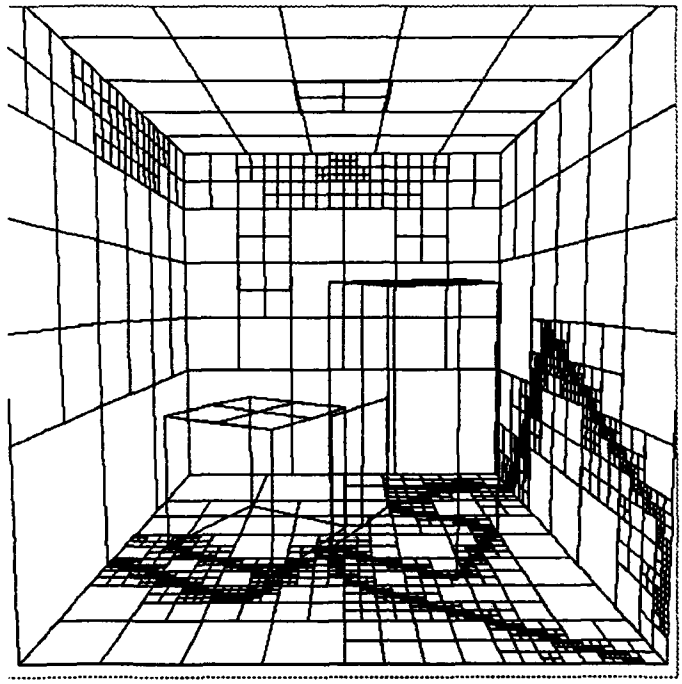

Figure 5.7. Cornell room showing subdivision after 6 levels.

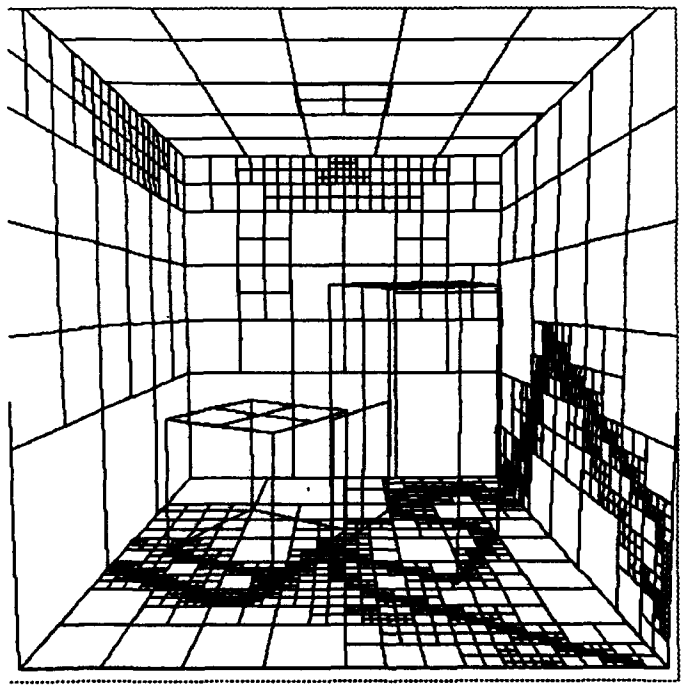

Figure 5.8. Cornell room showing subdivision after 7 levels.

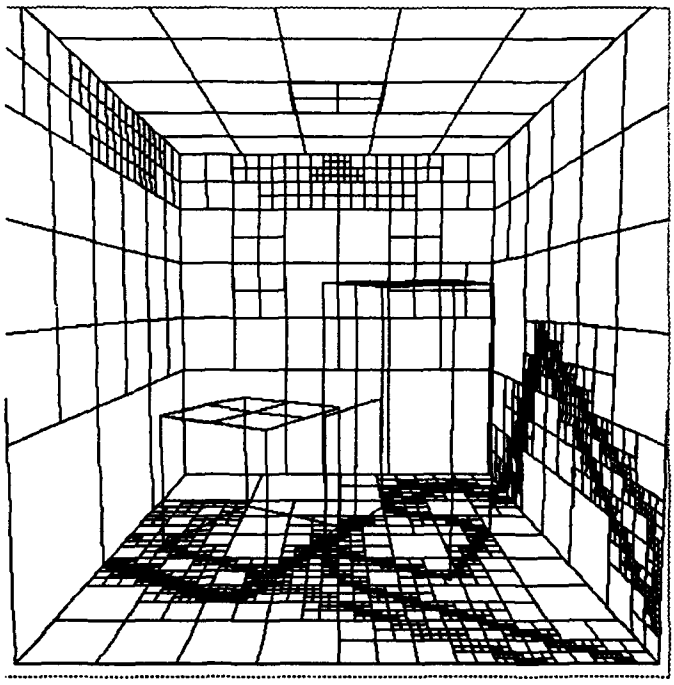

Figure 5.9. Cornell room showing subdivision after 8 levels.

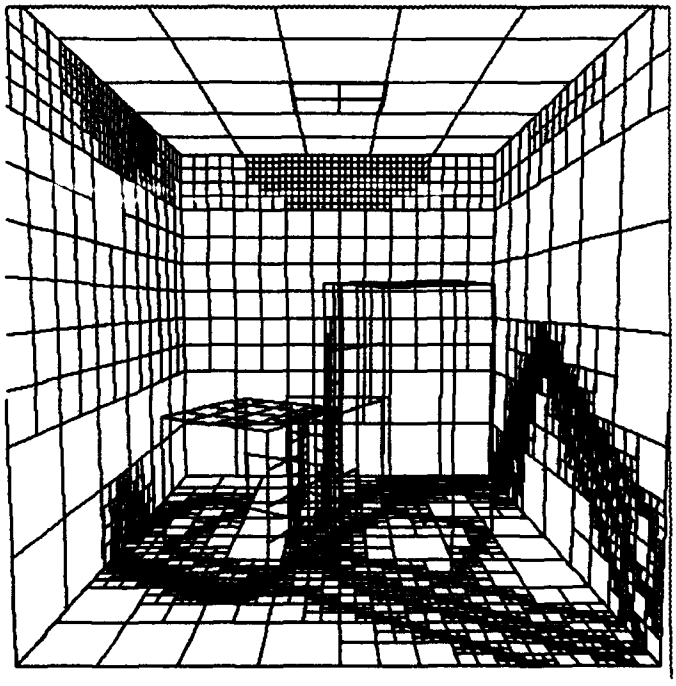

Figure 5.10. Cornell room showing final subdivision of elements.

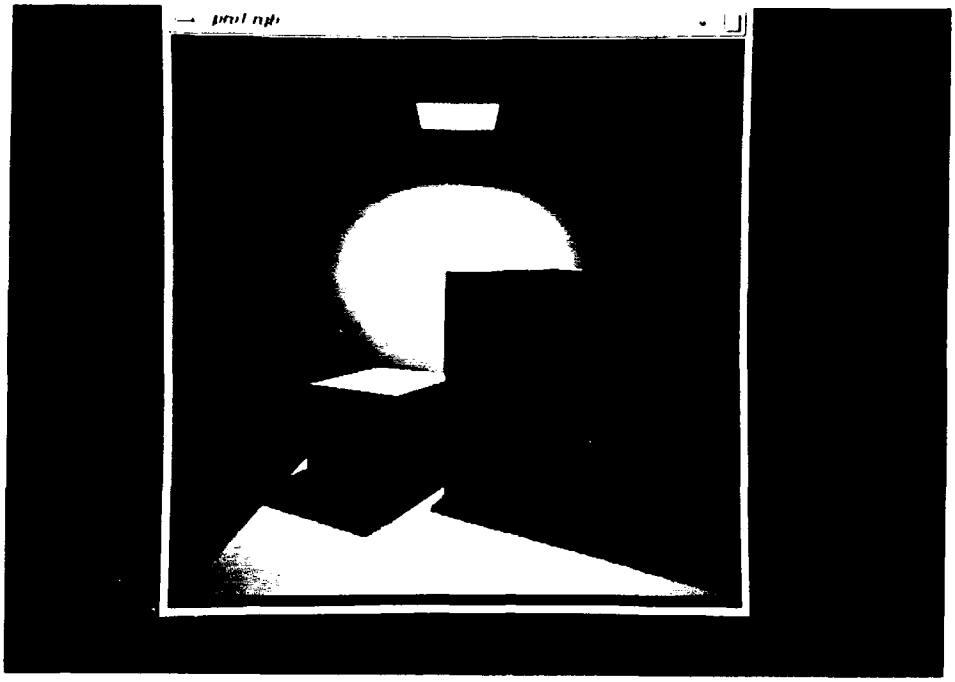

Figure 5.11. Cornell Room after one shooting patch.

greatest amount of unshot radiosity to be the next shooting patch. This allows for a good approximate solution after only a few shooting patches with changes becoming subtler with each remaining pass. Figures 5.11 through 5.15 show this progression for a simple scene.

5.1.3 Viewing the Image Once the radiosity solution is completed, the image is stored in a file for later viewing. The program *radview* is the tool for viewing the image. A full description of how to run *radview* is given in Appendix C. This program enables the user to view the image, changing the view point and reference point in real time. Figures 5.16 through 5.19 show a model of the interior of AWACS from different perspectives.

5.2 Directional Diffuse Radiosity

The directional diffuse radiosity program *extrad* goes beyond the capabilities of a standard radiosity renderer and renders accurate specular as well as diffuse reflections. It

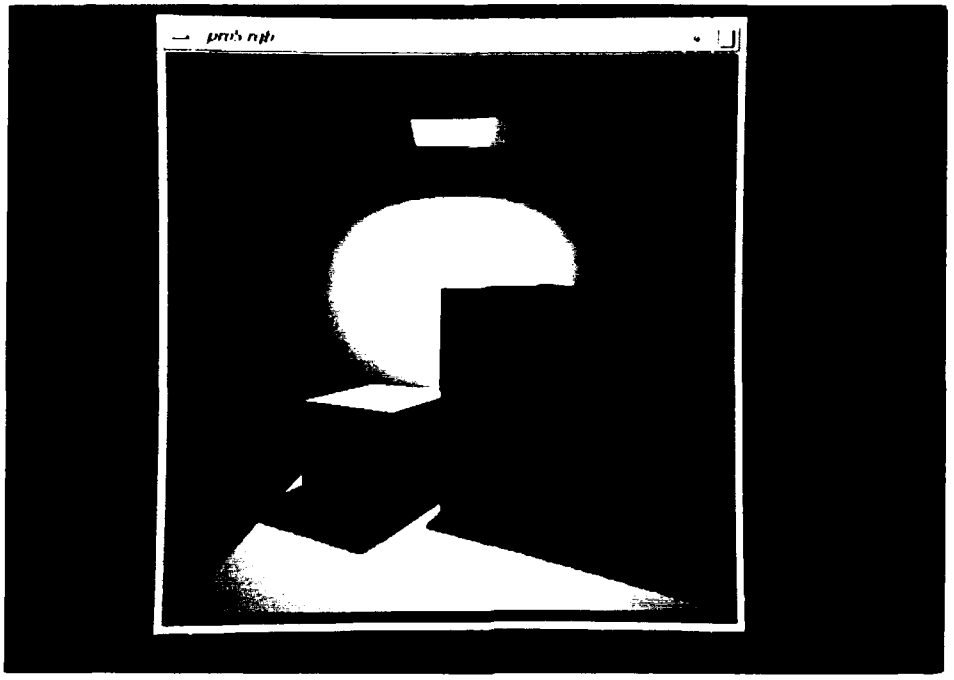

Figure 5.12. Cornell Room after five shooting patches.

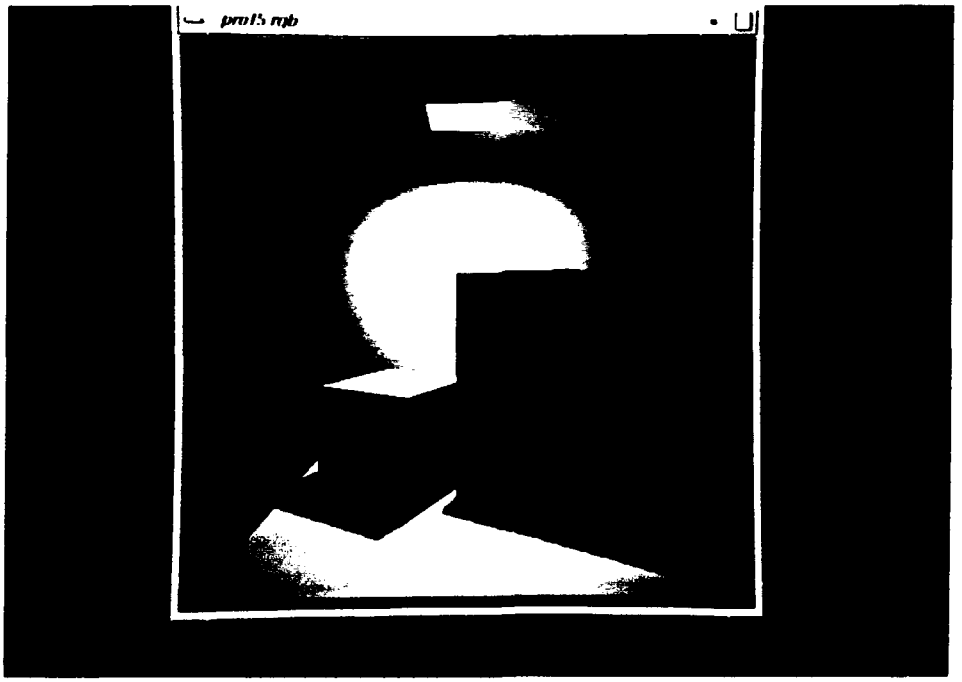

Figure 5.13. Cornell Room after 15 shooting patches.

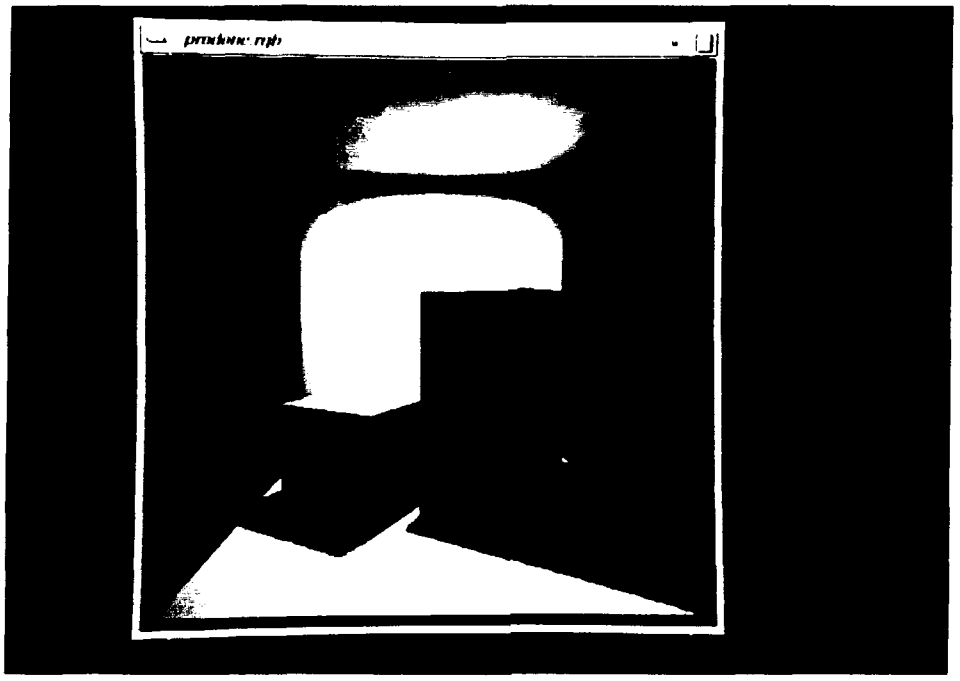

Figure 5.14. Cornell Room after convergence (63 patches shot).

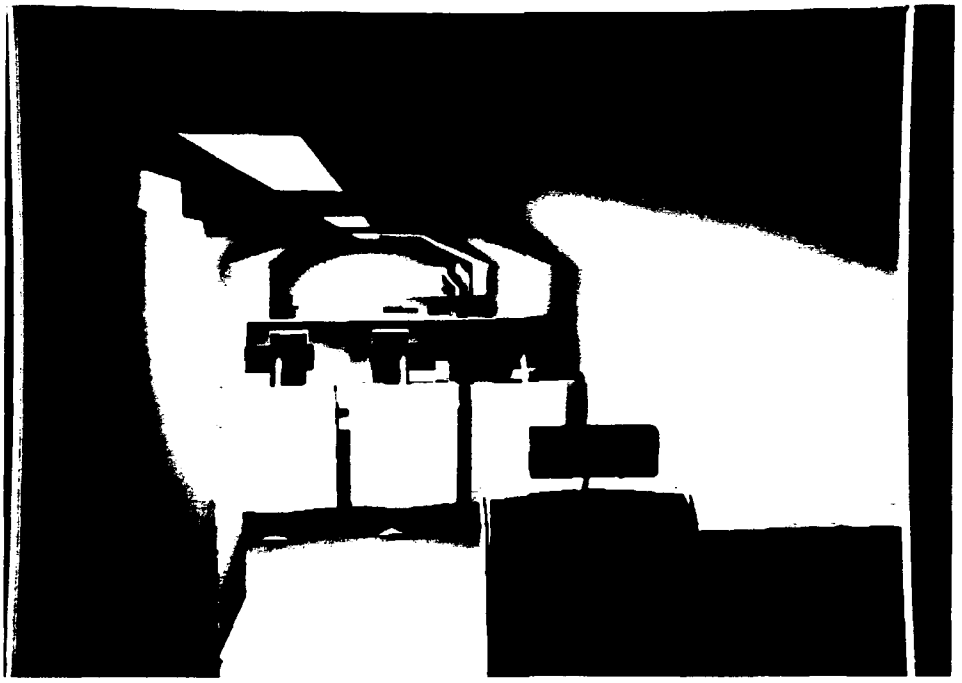

Figure 5.15. Interior of AWACs aircraft as rendered by *prorad*.

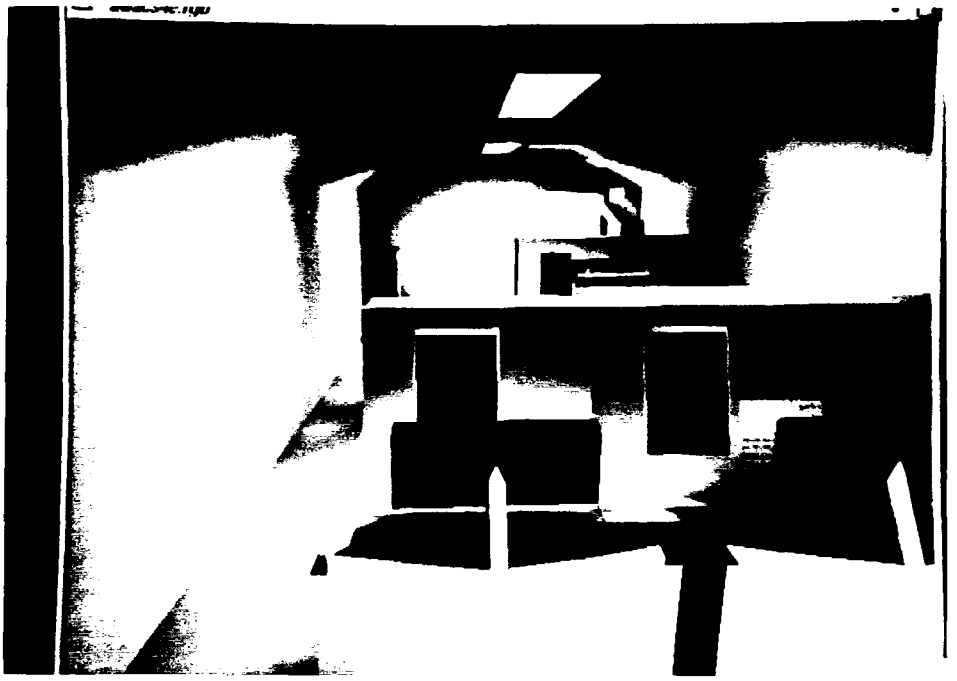

Figure 5.16. Closer look at two operator consoles

Figure 5.17. Details of AWACs chairs

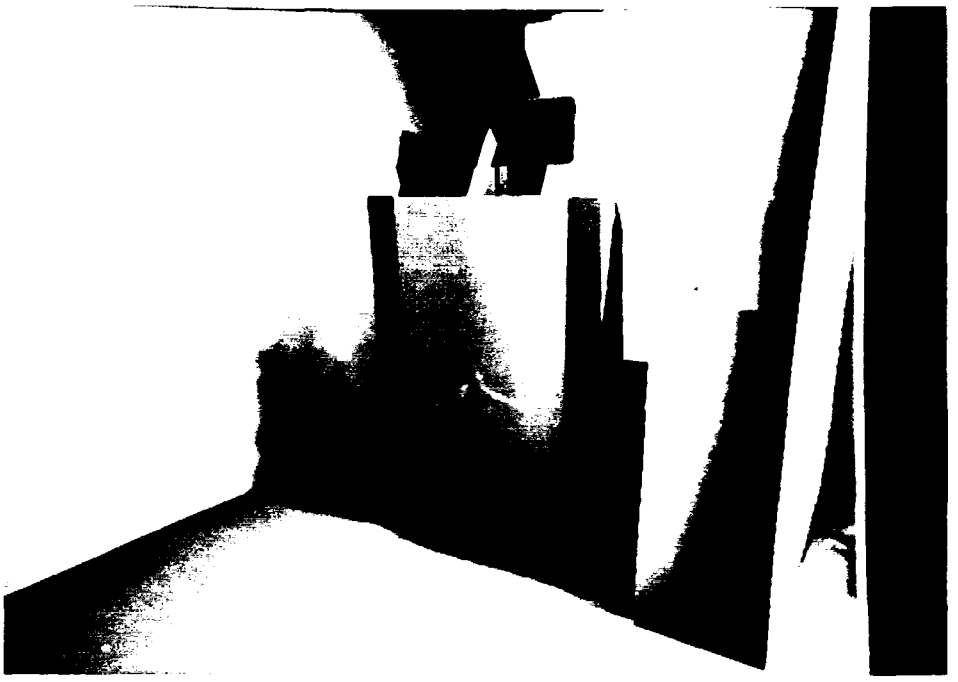

Figure 5.18. Commanders console

surface. The input geometry file is specified by the user as it is in the diffuse case and the file has the same format (as in Appendix B). Instructions for using *extrad* are found in Appendix H.

This program now models all mechanisms of light transport. These are: diffuse to diffuse, specular to diffuse, diffuse to specular, and specular to specular. Examples of images rendered with *extrad* are shown in Figures 5.19 to 5.24.

The primary difference between this system and what has been developed elsewhere is the fact that *extrad* uses hardware Gouraud shading to display the final image instead of following the radiosity calculations with a ray-tracing pass to do the final image display.

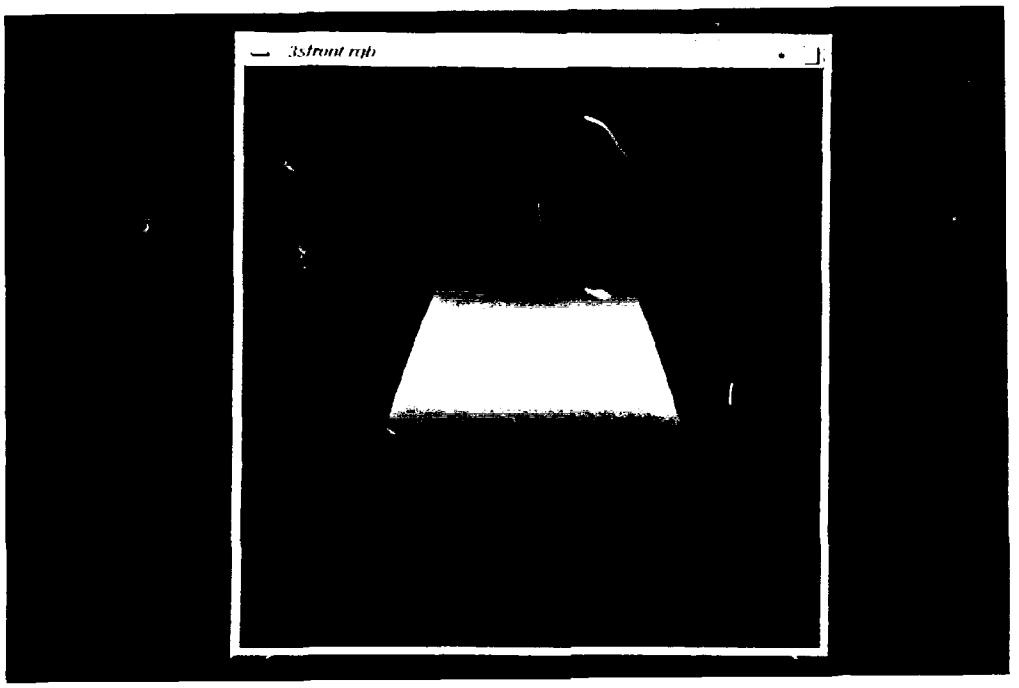

Figure 5.19. Image showing slight specular highlights(α = .15) from front (viewer is in direction of reflection).

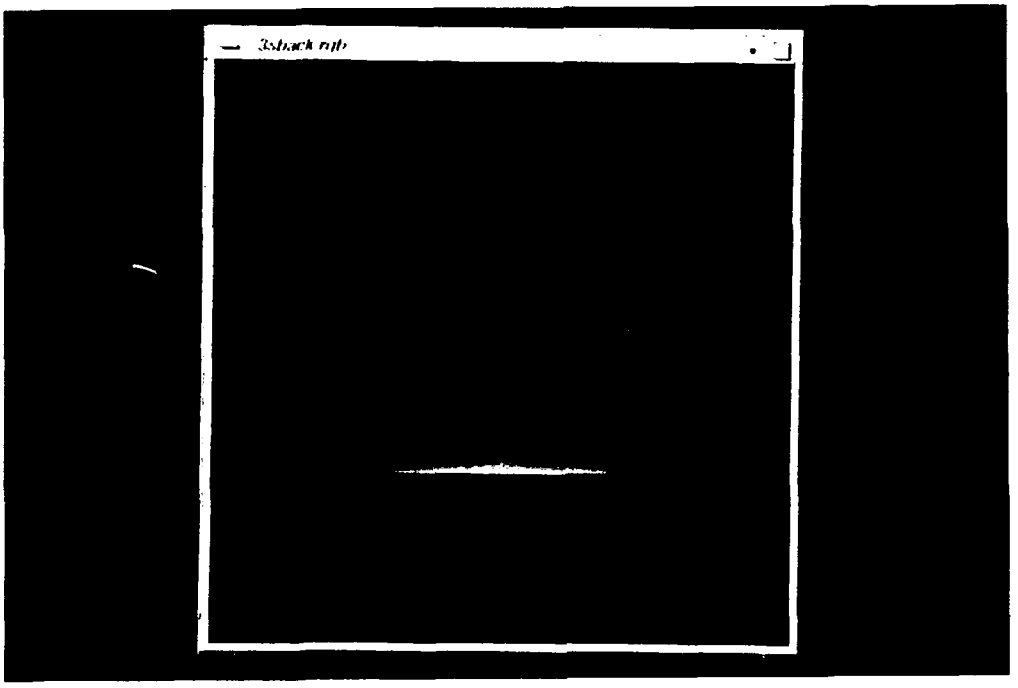

Figure 5.20. Image showing slight specular highlights(α = .15) from back (viewer is at light source).

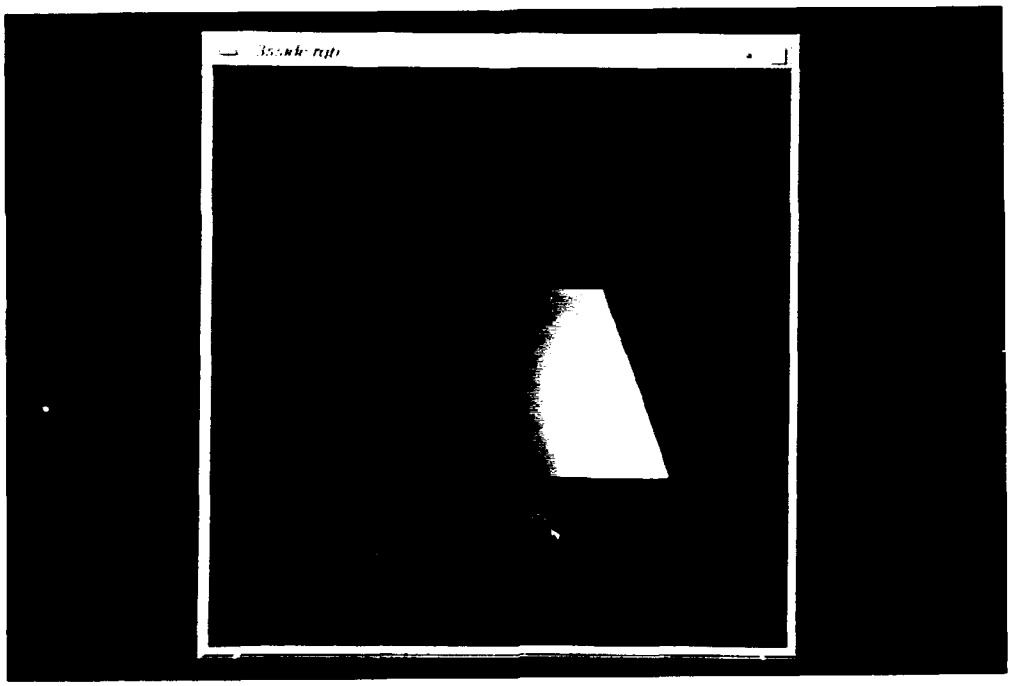

Figure 5.21. Image showing slight specular highlights($\alpha = .15$) from side.

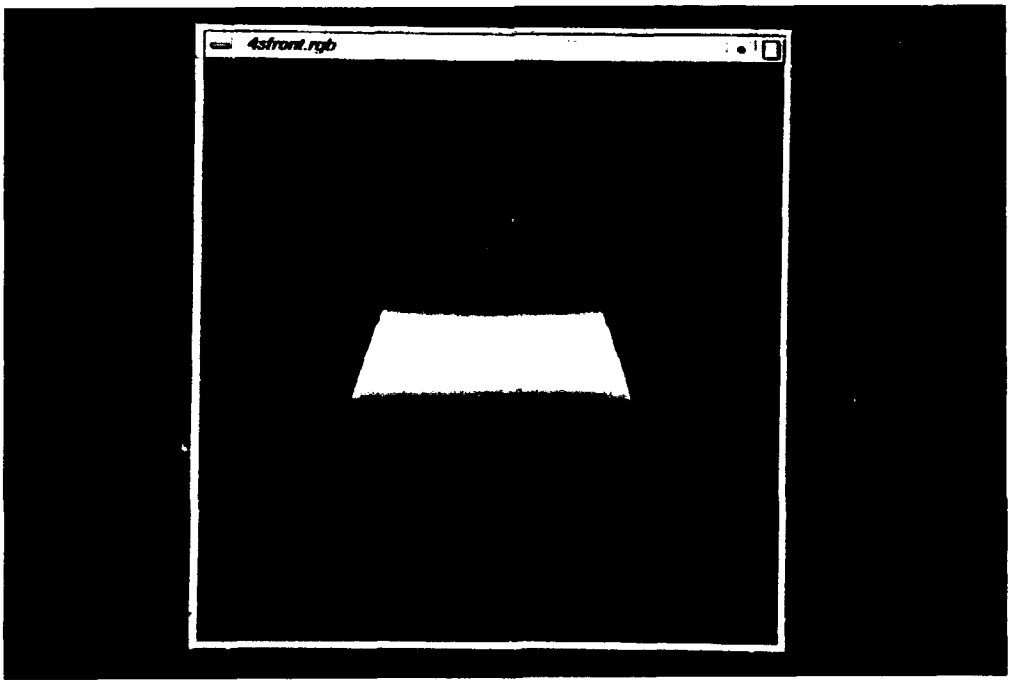

Figure 5.22. Image showing tighter specular highlights ($\alpha = .10$) from front.

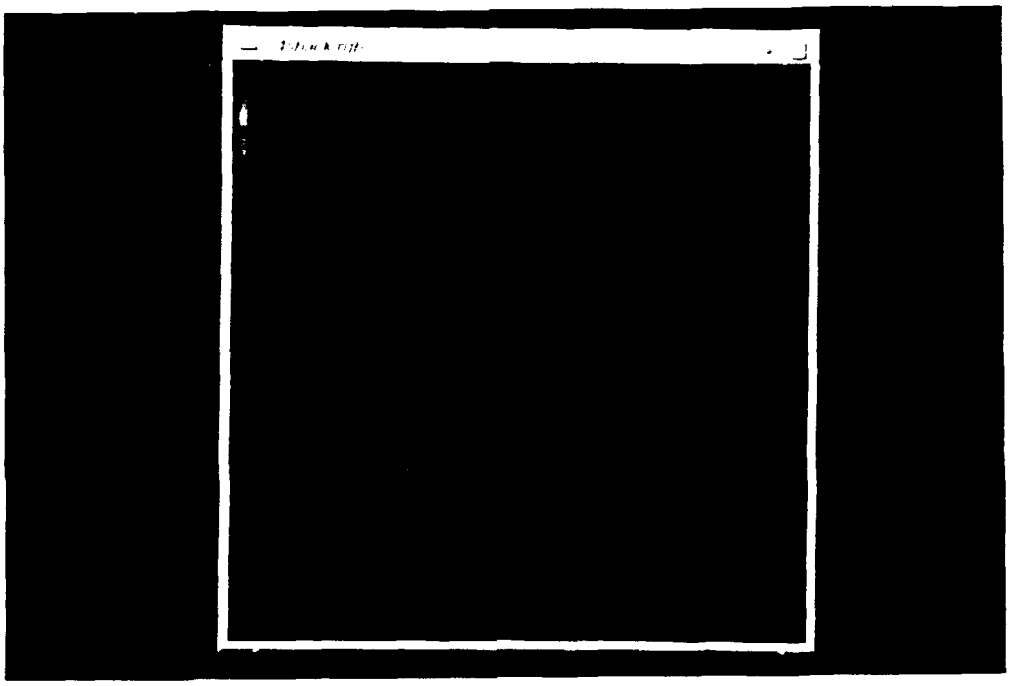

Figure 5.23. Image showing tighter specular highlights ($\alpha = .10$) from back.

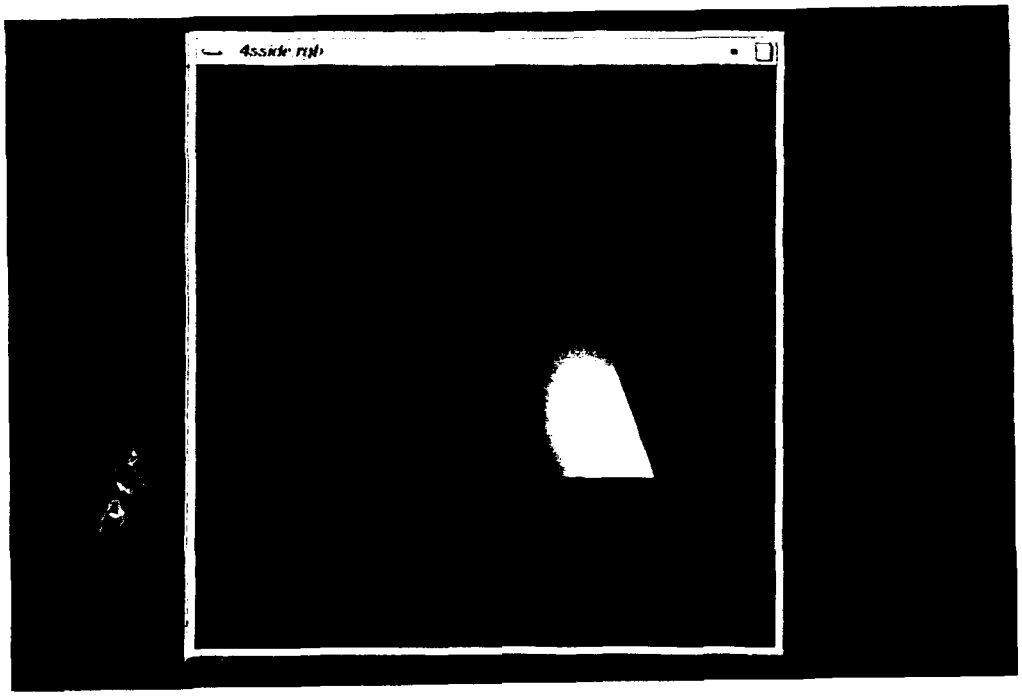

Figure 5.24. Image showing tighter specular highlights ($\alpha = .10$) from side.

5.3 Conclusion

Both *prorad* and *extrad* give AFIT a good radiosity capability that is not commercially available. From a relatively simple input description, the capability now exists to create realistically shaded images that model all methods of light transport.

VI. Recommendations for Future Research

This research provided two very good products that bring AFIT up to date in the radiosity field. These are a diffuse radiosity renderer that uses ray traced form factors and automated meshing, and an extended radiosity renderer that takes specular effects into account. While these capabilities are very good, there is still much that can be done.

6.1 Meshing Algorithms

Better ways of meshing hold some of the greatest promise for improving the radiosity method. Ideally, an environment should be divided into as few patches as necessary to give the required detail. It is also desirable to have all of the meshing done in an automated way so that less is required of the user.

There are a variety of areas that should be looked into in the area of meshing. These include:

1. Looking at the BSP-tree techniques as described by Campbell (5).
2. Examining other methods of discontinuity meshing (23).
3. Taking specular highlights into account when determining what areas to mesh more finely.
4. Finding a good way to determine which patches affect the way an environment should be meshed. The current implementation only uses the patches that are light sources to determine meshing requirements in the interest of rendering speed. This neglects meshing in areas that are in shadow of the light source but may be illuminated by some other bright surface.
5. Use an adaptive meshing technique where areas of high gradient are meshed more finely and elements with low gradient are combined with their neighbors to decrease the element count where the finer meshing is unnecessary.

6.2 Source Sampling

With the current implementation, shooting patches are sampled only once. This is good for speed but does not model shadows very accurately. Wallace et. (32) discuss some other ways of sampling the source patch that should be considered. A jittered sampling of the source that adapts the number of samples taken to the specific source would be a good approach.

6.3 Bi-directional Reflectance Distribution Functions

The area of BRDFs is also an area where further research is needed.

The most immediate need in this area is an accurate and easy way to get the BRDFs into the program as spherical harmonic coefficients. Right now it is difficult both to find accurate BRDF descriptions and to get those descriptions into the coefficients.

The next shortcoming of the current implementation is the fact that all of the BRDFs must be isotropic. This gives some pleasing specular effects, but many materials in the real world exhibit anisotropic reflection characteristics. This would be a good capability to add and is discussed in (34).

Another enhancement along these same lines would be to allow non- diffuse light sources. The software is already set up to handle such sources, but getting the intensity characteristics accurately loaded into the program is a challenge.

6.4 Ray Tracing

The current implementation uses hardware Z-buffer and Gouroud shading to display the image on the screen. This is done in the interest of rendering speed, but neglects the perfectly specular effects that could easily be added. This could be done by adding a simple ray tracing second pass as described in (27).

6.5 Other Enhancements

Another relatively straight forward extension would be the addition of a texture mapping capability. This would involve mapping the radiosity at the vertices that are to be texture mapped to an intensity value, and then doing standard texture mapping.

One further enhancement that would be of potential benefit is to parallelize the software to take advantage of a multi-processor system. Given a shooting patch, it would be very easy to separate the receiving vertices across multiple processors to calculate the radiosity.

The current version of *extrad* does not store the image into an external file for later use. This would be a good feature to add. This would also require developing a viewer to see the image. This viewer could be modelled after the display code that shows the image after it is rendered in *extrad.*

6.6 Conclusions

This research and thesis cover developing radiosity software. Since radiosity is such a new field, most of the existing literature only covers theoretical issues, but does little in the way of helping someone implement a functional radiosity renderer. This is especially true in the case of extended radiosity. This research works through the problems developing a radiosity system and the key issues are addressed in this document.

The biggest stumbling block for the extended radiosity developer is representing the Bi-Directional Reflectance Distribution Functions (BRDFs). This thesis shows a new way of representing these BRDFs that is much easier to incorporate. This stems from the fact that the BRDFs are easier to store, and require no complicated double integrals to calculate.

Appendix A. *Format for Display Parameter File*

- Line 1: Comment
- Line 2: *eye* < *x* > < *y* > < *z* > *lookat* < *x* > < *y* > < *z* > (eye and lookat positions)
- Line 3: *up* < *x* > < *y* > < *z* > (up vector)
- Line 4: *fovx* < *a* > *fovy* < *b* > (field of view in x and y)
- Line 5: *near* < *a* > *far* < *b* > (Near and Far clipping planes)
- Line 6: *xres* < *a* > *yres* < *b* > (Image resolution in x and y)
- Line 7: *diameter* < *a* > (Approximate Image Diameter)
- Line 8: *ambient* < 0|1 > (1 for ambient displayed)
- Line 9: *threshold* < *a* > (ratio of shoot patch energy to total energy times 1000)

Appendix B. *Format for Geometry File*

The input geometry format is the standard *AFIT Geom* format with the addition of a few parameters. These new parameters allow the user to specify: how finely a polygon should be divided into patches; which polygons are light sources; and also the number of the material file that corresponds to the surface type. Below is a line by line description of the file format; parameters that are not part of the *AFIT Geom* format are in bold type.

- Line 1: < *textstring* > (Informative title line)
- Line 2: [ccw] [cw] [purge] [nopurge] (Attribute selection line)
- Line 3: points < *# of points* > patches < *# of patches* > attributes < *# of attributes* > (Object component line count)
- Line 4 to 3 + < *# of points* >: < x >< y >< z > normal < $\vec{N}_x$ >< $\vec{N}_y$ >< $\vec{N}_z$ > (Control point lines)
- Line 3 + < *# of points* > +1 to
  3 + < *# of points* > + < *# of patches* >: 4 < *point index* 1 > ... < *point index* 4 > attribute < *# of attribute* > **patch** < *# of patches/polygon* > (Polygon lines)
- Line 3 + < *# of points* > + < *# of patches* > +1 to
  3 + < *# of points* > + < *# of patches* > + < *# of attributes* >: color < r >< g >< b > **[emit** < r >< g >< b >**]** **[material]** < *material #* > (Shading attribute lines)

Appendix C. *Using radview*

To execute *radview*, type the following from the command line:

radview < *filename* > [up-axis]

This allows the user to specify where the window will be open and how large it will be. up-axis is y, -y, z, or -z and is optional. The default is y.

C.1 Mouse and Keyboard Inputs

- F1 - Open the help window which contains this information.
- F2 - Close the help window.
- LEFT MOUSE - Hold and move mouse to change the azimuth and elevation of the viewpoint.
- RIGHT MOUSE - Menu selection for manual or automatic mode.
- UP ARROW - Move elevation of viewpoint up one degree.
- DOWN ARROW - Move elevation of viewpoint down one degree.
- LEFT ARROW - Move azimuth of viewpoint left one degree.
- RIGHT ARROW - Move azimuth of viewpoint right one degree.
- z - zoom in with constant reference point.
- x - zoom out with constant reference point.
- a - move viewpoint and reference point in direction of line of sight.
- s - move viewpoint and reference point in opposite direction of line of sight.
- r - reset all parameters to default view.
- ESC - exit *radview*.

Appendix D. *Using* prorad

To execute ***prorad***, type the following from the command line:

prorad < *-g geomfile* > < *-d displayfile* > [*-o outfile*] [*-w*] [*-s*] [*-t*] [*-n*]

OPTIONS

-g - Specify the input geometry. Format for this file is described in Appendix B.

-d - Specify the display format. This describes how the image is to be viewed. Format for this file is described in Appendix A.

-o - Specify the name of the file to store the image for later viewing. If this option is not specified, the image will not be saved.

-w - Specifies wireframe image is to be drawn rather than shaded image. Has no effect without *-s* option.

-s - Specifies image is to be shown while it is being rendered. Default is not shown.

-t - If this option is set, polygon intersection is not checked and no shadows are rendered. This is useful for testing as images are rendered very quickly.

-n - Specifies normals are to be drawn in. This option should only be selected with *-w*.

Appendix E. *Mathematica Program Listing*

```
f[x_] := Exp[-(1-x)/(alpha^2*(1+x))]

alpha = N[15/100,30]

a[k_Integer?NonNegative] := a[k] =
        NIntegrate[(2k+1)*f[mu]*LegendreP[k,mu]/2,
        {mu,-1,1},WorkingPrecision -> 20]

L = 10

Table[a[j],{j,0,L}] // TableForm
```

Appendix F. *Reflectance File Format*

- Line 1: $lval < L > Ks < Ks > Kd < Kd >$
- Line 2: $< a_0 >$
- Line 3: $< a_1 >$
- Line 4: $< a_2 >$

  ⋮

- Line L+2: $< a_L >$

Appendix G. *Expanded BRDF Derivation*

The first step in deriving a form for the BRDFs is to come up with a model for the surface reflectivity. For this research, a reflectivity that depends on θ^{in}, θ, and ϕ, namely $\rho(\theta^{in}, \theta, \phi)$ is needed. There are a variety of ways to get these ρ values. The most accurate would be to actually measure the reflectivity of different surfaces, but storage requirements and lack of adequate equipment make this choice impractical. The other alternative is to come up with a model that approximates ρ. This is the approach used for this research.

One such model was presented by Greg Ward at SIGGRAPH '92 (33). This model is:

$$\rho(\theta^{in}, \theta, \phi) = \frac{\rho_d}{\pi} + \rho_s \cdot \frac{1}{\sqrt{\cos\theta^{in} \cos\theta}} \cdot \frac{\exp[-\tan^2\delta/\alpha^2]}{4\pi\alpha^2} \tag{G.1}$$

where:

- ρ_d is the diffuse reflectance
- ρ_s is the specular reflectance
- δ is the angle between vectors $\vec{n}$ and $\vec{h}$ shown in Figure G.1
- α is the standard deviation (RMS) of the surface slope

While this equation does model the surface characteristic desired, it does not have all the properties necessary to easily adapt to calculating the spherical harmonic coefficients. This research uses a model similar to the Ward model but with better symmetry characteristics that make it a better choice. The following approach and analysis was provided by Dr. Kirk Mathews (24).

Suppose α is independent of θ^{in}:

$$\rho = C_R \exp\left[\frac{-\tan^2\delta}{\alpha^2}\right] \tag{G.2}$$

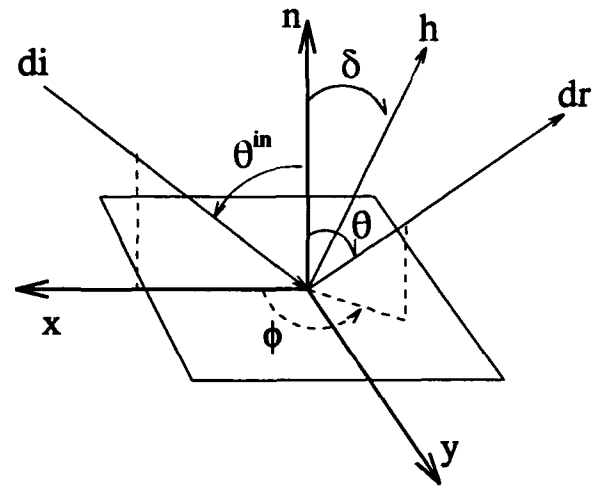

Figure G.1. Angles and vectors used in reflectance equation. $\vec{i}$ is the direction of incoming light, $\vec{r}$ is the direction of reflection, $\vec{h}$ is the halfway vector between $\vec{i}$ and $\vec{r}$.

where C_R is a normalization constant to be determined later. $\delta = \theta_0/2$ for $\phi = 0$ and for $\phi = \pi$, so approximate ρ as:

$$\rho = C_R \exp\left[\frac{-\tan^2(\theta_0/2)}{\alpha^2}\right] \tag{G.3}$$

and

$$\tan(\theta/2) = \frac{\sin\theta_0}{1 + \cos\theta_0} = \frac{\sqrt{1-\mu_0^2}}{1+\mu_0} \tag{G.4}$$

where $\mu_0 = \cos\theta_0 = \vec{\Omega} \bullet \vec{\Omega}'$ and $\vec{\Omega}$ is a unit vector in direction photon scatters from surface, while $\vec{\Omega}'$ is the direction it would scatter if surface were a perfect mirror.

Bring this together gives:

$$\rho(\mu_0) = C_R \exp\left[\frac{-(1-\mu_0^2)}{(\alpha(1+\mu_0))^2}\right] \tag{G.5}$$

After further simplification:

$$\rho(\mu_0) = C_R \exp\left[\frac{-(1-\mu_0)}{\alpha^2(1+\mu_0)}\right] \tag{G.6}$$

This can be approximated with a truncated Legendre polynomial expansion:

$$\rho(\mu_0) \approx C_R \sum_{l=0}^{L} a_l P_l(\mu_0) \tag{G.7}$$

The orthogonality of the Legendre polynomials can be used to find the a_l values:

$$\int_{-1}^{1} \frac{2l+1}{2} P_l(\mu_0) P_{l'}(\mu_0) d\mu_0 = \delta_{ll'} \tag{G.8}$$

$$\int_{-1}^{1} \frac{2l+1}{2} \rho(\mu_0) P_l(\mu_0) d\mu_0 = a_l \tag{G.9}$$

as shown below:

$$= \int_{-1}^{1} \frac{2l+1}{2} [\sum_{l'=0}^{L} a_{l'} P_{l'}(\mu_0)] P_l(\mu_0) d\mu_0 \tag{G.10}$$

$$= \sum_{l'=0}^{L} a_{l'} \int_{-1}^{1} \frac{2l+1}{2} P_{l'}(\mu_0) P_l(\mu_0) d\mu_0 \tag{G.11}$$

$$= \sum_{l'=0}^{L} a'_l \delta_{ll'} = a_l \tag{G.12}$$

This truncation could result in negative values for some μ_0 locations. This can be corrected by:

a) Add a small diffuse component which would make net result positive in all directions.

b) Otherwise "Fix-up" the coefficients to eliminate negative results.

It appears that the negative values are negligibly small, so choose option (a).

Normalization to the correct reflection fraction:

$$K_s = \int_0^1 d\mu \int_0^{2\pi} d\phi \; \rho(\mu', \mu, \phi) \tag{G.13}$$

$$\frac{K_s}{C_R} = \sum_{l=0}^{L} \int_0^1 d\mu \int_0^{2\pi} d\phi \left[a_l P_l(\mu') P_l(\mu) + \sum_{m=0}^{l} 2 \frac{(l-m)!}{(l+m)!} P_l^m(\mu') P_l^m(\mu) \cos(m\phi) \right] \tag{G.14}$$

$$= \sum_{l=0}^{L} \int_0^1 d\mu \int_0^{2\pi} d\phi \left[a_l P_l(\mu') \right] +$$

$$\sum_{l=0}^{L} \sum_{m=0}^{l} 2 \frac{(l-m)!}{(l+m)!} \int_0^1 d\mu \left[\int_0^{2\pi} d\phi \cos(m\phi) \right] P_l^m(\mu') P_l^m(\mu) \tag{G.15}$$

$$= \sum_{l=0}^{L} a_l P_l(\mu') \int_0^1 d\mu P_l(\mu) \int_0^{2\pi} d\phi \tag{G.16}$$

$$\frac{K_s}{C_R} = 2\pi \sum_{l=0}^{L} a_l P_l(\mu') C_l \tag{G.17}$$

where $C_l = \int_0^1 d\mu P_l(\mu)$, which can be obtained once and tabulated. This now gives the spherical harmonic expansion terms:

$$C_R(\mu') = \frac{K_s}{2\pi \sum_{l=0}^{L} a_l P_l(\mu') C_l} \tag{G.18}$$

$$b_{l,0}(\mu') = C_R(\mu') a_l P_l(\mu') \tag{G.19}$$

$$b_{l,0}(\mu') = C_R(\mu') 2 a_l \frac{(l-m)!}{(l+m)!} P_l(\mu') \tag{G.20}$$

Appendix H. *Using* extrad

To execute ***extrad***, type the following from the command line:

extrad < *-g geomfile* > < *-d displayfile* >

OPTIONS

-g - Specify the input geometry. Format for this file is described in Appendix B.

-d - Specify the display format. This describes how the image is to be viewed. Format for this file is described in Appendix A.

Bibliography

1. Arvo, James and David Kirk. "Particle Transport and Image Synthesis," *Computer Graphics*, *24*:63–66 (August 1990).
2. Arvo, James (Editor). *Graphics Gems II*. San Diego, California: Academic Press, 1991.
3. Baum, D. R., et al. "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors," *Computer Graphics*, *23*:325–334 (July 1989).
4. Campbell III, A. T. "Advanced Topics in Radiosity Meshing Algorithms." In *ACM SIGGRAPH - Course Notes - Radiosity*, pages 109–124, 1992.
5. Campbell III, A. T. and Donald S. Fussell. "Adaptive Mesh Generation for Global Diffuse Illumination," *Computer Graphics*, *24*:155–164 (August 1990).
6. Chen, Shenchang Eric, et al. "A Progressive Multi-Pass Method for Global Illumination," *Computer Graphics*, *25*:165–174 (July 1991).
7. Cohen, Micheal F. "Basic Radiosity Formulation for Diffuse Reflections." In *ACM SIGGRAPH - Course Notes - Radiosity*, pages III–1 to III–18, 1989.
8. Cohen, Micheal F., et al. "A Progressive Refinement Approach to Fast Radiosity Image Generation," *Computer Graphics*, *22*:75–84 (August 1988).
9. Cohen, Micheal F. and Donald P. Greenberg. "The Hemicube - A radiosity Solution for Complex Environments," *Computer Graphics*, *19*:31–40 (July 1985).
10. Cohen, Micheal F., et al. "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, *6*:26–35 (March 1986).
11. Cook, Robert L., et al. "Distributed Ray Tracing," *Computer Graphics*, *18*:137–145 (July 1984).
12. Cook, Robert L. and Kenneth E. Torrance. "A Reflectance model for Computer Graphics," *Computer Graphics*, *15*:307–316 (August 1981).
13. Courant, R. and D. Hilbert. *Methods of Mathematical Physics*. New York: Interscience Publishers, Inc., 1953.
14. Foley, et al. *Computer Graphics, Principles and Practice*. Reading, Massachusetts: Addison-Wesley, 1990.
15. Gillett, Philip. *Calculus and Analytic Geometry*. Lexington, Massachusetts: D. C. Heath and Company, 1981.
16. Goral, Cindy M., et al. "Modeling the Interaction of Light Between Diffuse Surfaces," *Computer Graphics*, *18*:231–222 (July 1984).
17. Hall, Roy A. and Donald P. Greenberg. "A Testbed for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, *3(10)*:10–20 (November 1983).
18. Hanrahan, Pat, et al. "A Rapid Hierarchical Radiosity Algorithm," *Computer Graphics*, *25*:197–206 (July 1991).

19. He, Xiao, et al. "A Comprehensive Model for Light Reflection," *Computer Graphics*, *25*:175–186 (July 1991).

20. Hill, Francis. *Computer Graphics*. Reading, Massachusetts: Addison-Wesley, 1990.

21. Immel, David S. and Micheal F. Cohen. "A Radiosity Method for Non-Diffuse Environments," *Computer Graphics*, *20*:133–142 (August 1986).

22. Kajiya, J. T. "The Rendering Equation," *Computer Graphics*, *20*:143–150 (July 1986).

23. Lischinski, Dani, et al. "Discontinuity Meshing for Accurate Radiosity," *IEEE Computer Graphics and Applications*, *17*:25–39 (November 1992).

24. Mathews, Kirk, "Private Communication," November 1992.

25. Maxwell, Gregory, et al. "Calculations of the Radiation Configuration Factor Using Ray Casting," *Computer Aided Design*, *18(7)*:371–37 (July 1986).

26. Phong, B.T. "Illumination for Computer Generated Images," *Communications of the ACM*, *18*:311–317 (June 1975).

27. Sillion, Francois, et al. "A Global Illumination Solution for General Reflectance Distributions," *Computer Graphics*, *25*:187–196 (July 1991).

28. Sillion, Francois and Claude Puech. "A General Two-Pass Method Integrating Specular and Diffuse Reflection," *Computer Graphics*, *23*:335–344 (July 1989).

29. Wallace, John. "Radiosity Extensions – Specular Reflections." In *ACM SIGGRAPH – Course Notes – Radiosity*, pages VI–A–1 to VI–A–11, 1989.

30. Wallace, John R. "Introduction to Meshing Algorithms for Radiosity." In *ACM SIGGRAPH – Course Notes – Radiosity*, pages 85–108, 1992.

31. Wallace, John R., et al. "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray-Tracing and Radiosity Methods," *Computer Graphics*, *21*:311–320 (July 1987).

32. Wallace, J.R., et al. "A Ray Tracing Algorithm for Progressive Radiosity," *Computer Graphics*, *23*:315–324 (July 1989).

33. Ward, Gregory J. "Measuring and Modelling Anisotropic Reflection," *Computer Graphics*, *26*:265–272 (July 1992).

34. Westin, Stephen H., et al. "Predicting Reflectance Functions from Complex Surfaces," *Computer Graphics*, *26*:255–264 (July 1992).

35. Whitted, Turner. "An improved Illumination Model for Shaded Display," *Communications of the ACM*, *23*:343–349 (June 1980).

36. Wolfram Research, "Mathematica computer program."

37. Yourdan, Edward. *Modern Structured Analysis*. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

Vita

Captain Richard Remington was born on 5 June 1963 in Corvallis, Oregon. He graduated from Woodburn High School in Woodburn, Oregon in 1981 and attended Oregon State University, graduating with a Bachelor of Science in Computer Engineering in June 1986. Upon graduation, attended Officer Training School at Lackland AFB, Texas, and received his commission. His first assignment was to Hanscom AFB, Massachusetts as a Command Center Software Engineer for the Command Center Evaluation System, Electronic Systems Division, Air Force Systems Command. He was responsible for prototyping and led the development of the manual input system of the Granite Sentry Program. After that he was lead engineer for the PRISM program, which is a system to develop a generic command center capability. He remained in this position until entering the School of Engineering, Air Force Institute of Technology, in May 1991.

Permanent address: 1575 Audrey Way
Woodburn, Oregon 97071

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE

A Radiosity Based Realistic Image Synthesis System

5. FUNDING NUMBERS

6. AUTHOR(S)

Richard L. Remington, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DARPA/ASTO
3701 North Fairfield Dr
Arlington, Va 22203-1714

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

Abstract

The Air Force Institute of Technology (AFIT) has not had the capability to generate photorealistic images. This thesis is a research effort to implement such a system at AFIT, and to apply knowledge from the particle transport community to this problem.

Investigation into various techniques for generating photorealistic images led to the conclusion that the radiosity method would serve as the foundation for development. This is because of the increased accuracy of the diffuse reflections with the radiosity method and the fact that the images produced with radiosity are view-independent.

Two distinct tools to generate photorealistic images are described. The first of these follows the progressive radiosity methodology using ray tracing for form factor calculation. This tool models only diffuse reflections. The second tool is also based on the radiosity method, but does away with the restriction that all surfaces be lambertian diffuse and models specular reflections as well.

The design and implementation of the specular radiosity tool uses continuous functions to represent the ***bidirectional reflectance distribution functions***, (BRDFs) and intensities. A new method of representing the BRDFs using a truncated Legendre polynomial expansion is presented.

14. SUBJECT TERMS			15. NUMBER OF PAGES
Radiosity, Form Factors			88
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)